LUDUS MANAGER

Antonio Domínguez Goñi
Juan Iñigo Monjas
Miguel Morata Moralejo

PROYECTO DE SISTEMAS INFORMÁTICOS,
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Curso 2012/2013

Directores:
Rubén Fuentes Fernández
María Victoria López López

AUTORIZACIÓN DE DIFUSIÓN Y UTILIZACIÓN

Los abajo firmantes, matriculados en la asignatura Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado durante el curso académico 2012-2013 bajo la dirección de la Dra. María Victoria López López del Departamento de Arquitectura de Computadores y Automática y el Dr. Rubén Fuentes Fernández del Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Antonio Domínguez Goñi

Juan Iñigo Monjas

Miguel Morata Moralejo

DEDICATORIA

Antonio:

A mi madre, a mi hermano, a mi novia, a Gonzalo y amigos por ayudarme siempre que lo he necesitado, por apoyarme para que consiga todo aquello que me proponga. Y sobre todo a José Antonio por darme fuerzas en cada momento.

No permitas que nadie diga que eres incapaz de hacer algo. Si tienes un sueño, debes conservarlo. Si quieres algo, sal a buscarlo.

En busca de la felicidad

Juan:

A toda mi familia, amigos y seres queridos, por haberme ayudado a ser como soy. A mis padres, por respetar y apoyar todas las decisiones que he tomado a lo largo de mi vida. No puedo olvidar mencionar a mis colegas más allegados, ya que la vida no sería lo mismo sin esos *ahí, isss y telepeh!* A los ingleses, por haber inventado el fútbol, algo que para mí, es mucho más que un simple deporte o entretenimiento, es una forma de vida. Y por supuesto, al Real Madrid, club de mis amores, por haberme otorgado tantas alegrías y momentos inolvidables.

Incluso un camino sinuoso, difícil, nos puede conducir a la meta si no lo abandonamos hasta el final.

Miguel:

A mi familia, por hacerme creer en mí y apoyarme para que consiga todo lo que me proponga; a mis amigos, por intentar darme siempre su apoyo y convertir días malos en buenos; a mis compañeros de proyecto, porque cuando no tiraba uno del carro lo hacía otro, por todas esas conversaciones por el grupo de Whatsapp con las que nos hemos desesperado pero sobre todo hemos reído. En resumen, a todos mis seres queridos, por estar ahí y hacerme la vida mejor, y porque una vez alguien dijo: "*en esta vida no hay que rodearse siempre de las mejores personas, si no de aquellas que mejor persona te hagan ser*". Y por último, no puedo olvidarme de una persona que destaca por encima de las demás, que siempre ha tratado de darme confianza y seguridad en mí mismo y ha llegado a conseguirlo muchas veces, que siempre se ha interesado por éste proyecto aunque no sepa lo que hacemos realmente o cómo lo hacemos, que me ha hecho luchar siempre por aquello que deseaba y ver el lado positivo de las cosas, y por supuesto me ha dado siempre un motivo para levantarme y disfrutar de la vida con intensidad. Gracias a todos.

La gente muere de sentido común, con una oportunidad perdida tras otra. La vida es el presente, el futuro no existe. Haz que tu vida arda siempre con la llama más intensa.

Oscar Wilde

AGRADECIMIENTOS

Agradecer a Victoria y a Rubén su dedicación y paciencia, donde gracias a sus consejos se ha podido realizar con éxito este trabajo. A Victoria por "aguantarnos" durante todo el año en la realización del proyecto y a Rubén por el gran esfuerzo que ha hecho durante y finalización del mismo.

También agradecer, a Rubén Díaz Ballesteros, alumno de Ingeniería Técnica de Sistemas en Informática, por ayudarnos en la elaboración de los sprites que se han usado en el proyecto.

Juan, Miguel y Antonio.

PRÓLOGO

Este proyecto denominado Ludus Manager es una idea original de los autores de esta memoria: Miguel, Juan y Antonio.

El grupo presentó originalmente varias ideas de proyecto, todas ellas novedosas e innovadoras. De entre ellas, Ludus Manager fue la que finalmente eligieron. Para ello realizaron un análisis informal previo tipo DAFO, donde exponían los pros y los contras para el desarrollo completo de cada propuesta. Finalmente propusieron el desarrollo de un juego competitivo basado en la gestión de una Ludus o escuela de Gladiadores en la antigua Roma. Su entusiasmo por finalizar el proyecto no ha decaído en ningún momento a pesar de los problemas encontrados en el camino. Todos ellos se han involucrado al máximo invirtiendo tantas horas de estudio y trabajo como han podido encontrar a lo largo del curso y han conseguido aplicar técnicas alternativas para conseguir optimizar el comportamiento del juego, concretamente técnicas de Inteligencia Artificial basadas en el uso de Lógica Borrosa.

Es una satisfacción que los alumnos de Sistemas Informáticos puedan proponer y desarrollar sus propias ideas y un deber por nuestra parte contribuir a la consecución de sus objetivos. Desde aquí les deseamos mucha suerte y les agradecemos haber confiado en nosotros para dirigir este proyecto y deseamos ver su juego en las pantallas de los smartphones durante mucho tiempo.

Rubén Fuentes y Victoria López

ÍNDICE

ÍNDICE DE ILUSTRACIONES

RESUMEN

Ludus Manager es una propuesta de juego innovadora, destinada a aquellos usuarios que deseen potenciar sus habilidades de gestión y administración, y que sobre todo, busquen algo más que un simple videojuego de lucha. Nuestro proyecto es un juego que se aleja de los cánones típicos en los que se basan los juegos actuales.

Ludus Manager se enmarca en el contexto de juegos competitivos donde se plantea un tipo de actuación: competición offline (máquina vs usuario).

La aplicación adentra al usuario en el mundo de la antigua Roma, donde tiene como finalidad, alcanzar el éxito como dueño de una escuela de gladiadores (denominado "lanista"). Para ello sentirá las responsabilidades que conlleva esa posición y el peso de las decisiones que tome. El usuario puede consultar la información de su escuela y la de sus gladiadores, pero además, bajo el pago de una cantidad de monedas, también puede ver la información de la escuela rival para que sus decisiones sean las correctas. El jugador tiene la opción de asignar entrenamientos específicos a sus luchadores, con la finalidad de mejorar sus habilidades tanto físicas (fuerza, destreza....) como estilo de combate (habilidad de un arma en una mano, lucha cuerpo a cuerpo…). También puede realizar combates para poner a prueba a sus gladiadores. Dichos combates son generados de manera predeterminada, especificando el tipo de combate, la recompensa, el número de combates que tiene el torneo y una descripción del torneo. El jugador dispone, una vez acabado el combate, de toda la información de dicho enfrentamiento, además de poder visualizar el combate en 2D y ver si sus decisiones han sido correctas o no. A medida que el jugador gana torneos, aumenta la cantidad de monedas de su escuela y la fama tanto de los gladiadores como la de la propia escuela. Cuando tenga suficiente número de monedas, el usuario puede mejorar su escuela con la adquisición de nuevas armas y gladiadores o la mejora de las instalaciones. También tiene la opción de vender aquellos gladiadores que vea necesario, porque no sea realmente bueno o porque necesite dinero para comprar otro gladiador mejor. Dicho proceso de venta, se hace por medio de un sistema de inferencia fuzzy.

El juego se ha desarrollado como un proyecto para el sistema operativo Android. Entre sus objetivos de diseño se cuenta que sea fácil de actualizar y de evolucionar según los requisitos que demanden los usuarios, como nuevos tipos de gladiadores o de armas, por ejemplo.

Palabras clave: Fuzzy**,** Android, Videojuego, Jugador

ABSTRACT

Ludus Manager can be described as an innovative gaming proposal, aimed at users who want to enhance their management and administrative skills, and above all, looking for something more than just a standard fighting game. Our project will move away from the typical standards on which currents games are based upon.

This game can be placed in the competitive game realm, where the player is faced with one choice: offline competition (user vs CPU).

The application, gives the user the opportunity to enter into the world of ancient Rome, where its purpose will be to succeed as the owner of a school of gladiators (known as ludi magister). To achieve this, he will feel the responsibilities that someone in such a powerful position deal with, where every decision made has a crucial outcome. The user can consult the information of the school or gladiators at any moment, and even more, by paying a fixed quantity of coins, he can also spy on the rival schools, giving him the chance to make more thoughtful decisions, to overcome future encounters with success. The player also has the option to assign specific training techniques to his fighters to improve their physical abilities(strength, resistance…) or battling styles(hand to hand combat, one hand weapon ability…). Battles between gladiators are also available. These fights are generated in a predetermined way, being the user able to specify various aspects of the battle, such as the type of battle, the reward, the number of rounds the gladiator will have to face, and a description of the tournament. After the fighting, the player will be able to access all of the information regarding these battles, and in addition to this, a recreation of the battle with a 2D graphics animation. This will help the user conclude whether or not right choices were made before the fight. As the player participates and wins tournaments, he will have access to more coins and the fame of both the school and its gladiators will also increase. With enough coins, the user will be able to improve the school with new weapons and gladiators, or upgrade its facilities. The user will also have the option of selling one or several gladiators, in order to make more money or just because the gladiator fighting skills are not good enough to stand up for upcoming challenges. The previously mentioned selling process is developed with a fuzzy inference system, where the value of each gladiator is thoroughly calculated.

The project is developed for the Android operating system. An easy modification and upgrading of the system, adapting to the users' requirements on demand, are its main design objectives.

Keywords: Fuzzy, Android, Game, Player.

CAPÍTULO 1. INTRODUCCIÓN

A finales de los años 90, los teléfonos móviles solo tenían la funcionalidad de poder establecer llamadas con otros dispositivos. Algunos fabricantes como Nokia decidieron ofrecer algún tipo de entretenimiento en sus teléfonos móviles, y para ello introdujeron pequeños juegos. Uno de los primeros fue Snake, el cual, aun siendo desarrollado con unos gráficos poco elaborados, sigue manteniendo la misma popularidad de entonces. Estos juegos fueron evolucionando y algunos ofrecieron la posibilidad de desbloquear niveles pagando o conectándose a Internet, para así poder seguir avanzando dentro del mismo. Hasta ese momento, los videojuegos en los dispositivos móviles estaban integrados en el teléfono y guardados en la memoria ROM del móvil. Sin embargo, con los teléfonos móviles programables, había una zona de memoria donde se podían grabar datos. Se podía utilizar un lenguaje de desarrollo como Java para crear aplicaciones, y con un cable USB instalarlas en dicho dispositivo. Los móviles fueron evolucionando con el paso del tiempo, y con ellos la memoria, la potencia y los lenguajes de programación que se podían usar.

1.1 Estudio del mercado de las aplicaciones Android.

Actualmente el mercado de aplicaciones para móviles está experimentando un gran crecimiento con cifras de ventas muy elevadas. Hoy en día existen aplicaciones de todo tipo, como videojuegos en 3D, streaming de videos o acceso a servicios de información.

El sector de los videojuegos para dispositivos móviles tiene un tamaño de 850 millones de dólares y se pronostica que alcanzará los 1.5 billones de dólares en 2014. Para dicho año, se estima que casi 95 millones de personas jugarán a algún videojuego en su dispositivo móvil, estudio realizado por Pro|Chile [1].

Desde el punto de vista comercial, la implementación y desarrollo que tiene un videojuego para los teléfonos móviles es menos costosa y arriesgada pero más rentable que la producción de videojuegos para consolas u ordenadores de sobremesa. La gran diferencia entre ambos tipos de videojuegos es que el desarrollo de un videojuego para un dispositivo móvil es un proyecto a corto plazo, lo que minimiza la inversión y permite tener ingresos rápidamente. Otro aspecto muy importante es la distribución de las aplicaciones, ya que en un teléfono móvil se puede realizar vía descarga desde el mismo dispositivo.

1.2 Proyecto a desarrollar

El videojuego que se ha desarrollado para este trabajo de fin de carrera tiene la finalidad de potenciar la capacidad de gestión y administración del usuario. Para ello, el jugador debe dirigir una escuela de gladiadores, llamadas “Ludus” en la antigua Roma, dentro de un mundo virtual que irá evolucionando y proporcionando nuevos retos y experiencias al jugador.

Una vez que el usuario se ha descargado la aplicación Android en su teléfono móvil, puede iniciar una nueva partida y comenzar a jugar. Se comienza con una escuela de gladiadores básica, a la que puede dar el nombre que quiera, y que no posee ningún gladiador. A partir de ahí, debe ir gestionando su nueva escuela para convertirla en una de las más poderosas de toda Roma.

La primera tarea de la que tiene que ocuparse el jugador es la de comprar gladiadores con los que formará su escuela. Esta tarea ya le exige tomar decisiones sobre cómo gastar el dinero limitado que posee en un luchador u otro según las habilidades de los mismos. Una vez comprados, debe diseñar un plan de entrenamiento específico para cada uno de sus luchadores si quiere ganar los combates que se le presenten durante el transcurso de la partida. Para ello tiene que estudiar bien sus características, asignándoles entrenamientos correctos para sacar el máximo provecho de sus habilidades.

El jugador también tiene la posibilidad de ir mejorando las instalaciones de su escuela hasta un límite definido por el nivel de la ciudad en la que esté. Las mejoras en la escuela elevan el nivel de ésta, permitiéndole así tener acceso a mejores tipos de entrenamientos, obtener equipamiento mejorado y más variado, o poder alojar a más gladiadores dentro de los muros de su ludus. Como objetivo a largo plazo del jugador, se le dará la opción de "mudarse" a otra ciudad, vendiendo su escuela y adquiriendo una en otra ciudad mejor, llevándose consigo a sus gladiadores, empleados y equipamiento adquirido. Cuanto más importante sea la ciudad en la que se encuentra la escuela, mejores instalaciones podrá construir, mejores gladiadores podrá reclutar, y a mayores retos se enfrentará, aunque las recompensas serán también mayores.

El progreso del videojuego se hace de manera semanal, y el jugador puede observar cómo evoluciona su escuela según sus decisiones. Cada cierto tiempo irán apareciendo eventos en los cuales el jugador puede o debe participar (dependiendo del tipo de evento), tales como torneos en la arena o combates de exhibición. Estos eventos le permiten ganar fama y reputación, tanto para su escuela como para sus gladiadores. Esto último influirá muchísimo en el devenir del videojuego, ya que cuanto más importante sea la escuela y más conocidos sean sus gladiadores, más probabilidades tiene el jugador de participar en combates importantes o recibir mejores recompensas.

Los combates se realizan y se calculan de manera automática, mediante una serie de algoritmos que determinan el resultado de cada asalto (unos 3 segundos de combate). Estos algoritmos tienen en cuenta diferentes variables, tales como las características y habilidades de cada gladiador, su equipamiento, su estilo de combate, y la "tendencia" que adquiere el combate. A fin de que los resultados no sean deterministas, se introduce una componente de aleatoriedad en el cálculo. Sin embargo, una vez iniciado el combate, éste discurre de manera automática, y el jugador sólo puede visualizar mediante animaciones los momentos más importantes de la lucha.

Al finalizar el combate, el usuario recibe un resumen del mismo. Sus gladiadores habrán recibido heridas de diversa consideración, y dependiendo de su gravedad, los gladiadores deberán ser atendidos por el médico de la escuela. El tiempo de recuperación de estos daños depende de su gravedad y del nivel del médico. Solamente en algunos casos especiales o en torneos de alto nivel puede ocurrir que se pierda un

gladiador a causa de su derrota. Todos los combates (oficiales o de exhibición) en los que participen los gladiadores les proporcionarán reputación.

Además de participar en combates, el jugador ha de contratar empleados que le ayuden y asesoren en diversas tareas. El entrenador será una de las piezas básicas de la escuela, ya que de él depende en gran parte que los gladiadores progresen con sus entrenamientos rápidamente. Otro empleado importante es el médico que influye en la salud y la velocidad de recuperación de las heridas de los luchadores.

1.3 Estructura de la memoria.

El resto de la memoria se organiza de la siguiente manera:

- *Capítulo 2- Estado del Arte*, donde se exponen las aplicaciones que hay en el mercado hoy en día y que son parecidas al proyecto que se va a desarrollar en este documento.

- *Capítulo 3- Especificación de Requisitos*. Se exponen todas las funcionalidades de la aplicación, los dispositivos que se han utilizado, etc.

- *Capítulo 4-Desarrollo de la aplicación*. Donde se muestra como se ha desarrollado la inteligencia artificial de la máquina, las librerías que se han utilizado para la parte gráfica y los algoritmos que se han desarrollado para que las distintas funcionalidades de la aplicación sean lo más reales posibles.

- *Capítulo 5-Técnicas especiales utilizadas*. Se expone el sistema de inferencia fuzzy, el cual, calcula el precio de un gladiador de la escuela del usuario según unas características del mismo.

- *Capítulo 6- Conclusiones y Trabajo Futuro*.

CAPÍTULO 2. ESTADO DEL ARTE

En el año 2009 aproximadamente, el sector de los videojuegos móviles se disparó con el famoso juego Angry Birds. Este juego se encuentra disponible para sistemas operativos como iOS y Android, y ha conseguido más de 200 millones de descargas en todo el mundo. Debido a este fenómeno, IDC (*International Data Corporation*) [2] realizó cálculos y prevé que en 2015 habrá 182.700 millones de descargas de aplicaciones al año.

Actualmente se puede elegir entre una gran variedad de juegos en el Android Market [3]. Entre los más atractivos para los usuarios se encuentran aquellos que recuerdan a los clásicos juegos de la historia del videojuego, y los que incorporan algún aspecto novedoso respecto a su competencia. Los juegos que hay disponibles en el Android Market pueden clasificarse en aquellos cuya descarga es gratuita y aquellos que son de pago. Lógicamente, los primeros suelen contar con un mayor número de descargas. Una tercera dimensión de clasificación es el tipo de juego, con las siguientes categorías principales:

- *Arcade*: consiste en la manipulación de uno o varios personajes del juego, donde se van superando una serie de niveles de dificultad y se desbloquean nuevas características del juego.

- *Carreras*: videojuegos ambientados en las competiciones de automóviles, motocicletas, etc. Por lo general no ofrecen una historia o argumento de juego.

- *Deportes*: basados en cualquier deporte y su reglamentación. Por lo general no ofrecen una historia o argumento de juego.

- *Habilidad*: exige una cierta habilidad por parte del jugador ante retos de lógica que se le propongan. A medida que el usuario va avanzando en el juego, los retos adoptan un nivel de dificultad mayor.

- *Estrategia*: se trata de una mezcla entre el estilo *Arcade* y *Habilidad*, ya que exige un nivel de habilidad y entendimiento de la mecánica del juego superior al de un videojuego arcade, pero se basa en los mismos principios de superación de fases y desbloqueo de nuevas características.

El videojuego que se expone en este trabajo de fin de carrera, llamado Ludus Manager, está dentro de la categoría *estrategia*, donde ya se encuentran videojuegos muy parecidos al desarrollado en este documento, como son algunos de los que se exponen a continuación. Estos videojuegos se han analizado en la fase primaria del proyecto con el fin de diseñar un juego competitivo en el mercado y un proyecto viable.

2.1 Spartacus The Game (Facebook)

Se trata de un emocionante juego inspirado en la exitosa serie Spartacus. Cada jugador toma el papel de un *lanista*, dueño de una escuela de gladiadores. Las casas de los jugadores están compitiendo continuamente para poseer el mejor ludus y los gladiadores más temibles. En la Fig.1 se puede ver el aspecto de tu ludus en el menú.

Como lanista, el jugador tiene una variedad de posibilidades a su disposición, como por ejemplo entrenar a sus gladiadores para mejorar sus habilidades, participar en torneos enfrentándose a gladiadores procedentes de cualquier ludus del mundo para ganar fama y oro, retar a sus amigos en combates de exhibición, o comprar equipamiento mejorado que haga a sus gladiadores aún más letales.

Fig. 1. Menú Principal de Spartacus The Game

Una de las desventajas es que a éste juego solo se puede acceder con conexión a internet, mediante Facebook. Esto es una limitación, ya que los usuarios no siempre tienen acceso a la red y por tanto el jugador no puede disfrutar en ciertos momentos del día del videojuego.

Por otro lado, se trata de un juego "en tiempo real", algo que, según numerosas opiniones de los jugadores habituales, ya no es un estilo de juego atractivo, y por tanto no será el estilo del videojuego que se desarrolla en este proyecto de fin de carrera.

Entre sus puntos a favor, destacan sobre todo su atractivo aspecto gráfico y su estilo de juego adictivo, mediante el cual el usuario va evolucionando a sus gladiadores y desbloqueando nuevas y mejores características según va avanzando.

2.2 Blood & Glory

La característica principal de este juego es que la intensidad del combate se vive en estado puro. El usuario puede equipar a su gladiador con todo el arsenal del que dispone, como armas letales y armaduras hechas a medida para luchar en todos los estilos de combate. El jugador tiene la posibilidad de realizar ataques especiales y combos para alcanzar la victoria. Un ejemplo de combate se puede ver en la Fig.2.

Fig. 2. Combate de Blood & Glory

La principal desventaja de *Blood & Glory* es que no requiere la toma de ninguna decisión por parte del usuario. El desarrollo del combate es demasiado repetitivo, donde los movimientos están predefinidos, y por tanto, no da opción al usuario a actuar libremente. Además se alcanza el final del juego demasiado pronto.

Los puntos positivos de este juego son claramente los combates en 3D y en primera persona, ya que son de una calidad gráfica excelente. Dispone de un modo de juego offline, es decir, no es necesario estar conectado para jugar, lo cual da la posibilidad de jugar en cualquier parte.

2.3 Spartacus: Legends

Se trata de un videojuego aún en desarrollo, para las plataformas PS3 y XBOX360, cuyo estilo de juego será exactamente el mismo que el estilo de juego del proyecto de fin de carrera que aquí se expone. El usuario deberá contratar gladiadores y equiparlos

con las mejores armas y armaduras que posea, para después participar en torneos y enfrentamientos contra los gladiadores de cualquier usuario del mundo que esté jugando en la misma plataforma que el jugador. Básicamente este videojuego se trata de una mezcla de todos los videojuegos vistos anteriormente.

Fig. 3. Combate de Spartacus: Legends

Como se ha dicho, reúne ventajas de todos los videojuegos antes citados. Posee una calidad gráfica excelente, da la opción al usuario de controlar a su gladiador durante el combate como puede verse en la Fig.3, y además no quita protagonismo a la parte de estrategia y gestión que, al igual que en este proyecto de fin de carrera, permiten ir mejorando las habilidades de los gladiadores y desbloqueando nuevas funcionalidades.

Sin embargo, también reúne varias de las desventajas que poseen los videojuegos antes citados, como la imposibilidad de jugar offline, o sin ir más lejos, el hecho de que solo se pueda jugar en dispositivos no portátiles, como son PS3 y XBOX360.

CAPÍTULO 3. ESPECIFICACIÓN DE REQUISITOS

3.1 Introducción

Este capítulo representa la especificación de requisitos del sistema. En los siguientes apartados definiremos de forma más precisa el proyecto Ludus Manager, el objetivo final de nuestro desarrollo, y todos los requisitos de partida que serán usados como base de su diseño y posterior implementación.

El objetivo principal de este proyecto es diseñar una aplicación móvil para ejecutar en el sistema operativo Android cuya finalidad es entretener al usuario mientras desarrolla su capacidad de gestión, adentrándole en el mundo de la Antigua Roma.

El juego que se está exponiendo en este trabajo de fin de carrera pertenece al tipo estrategia, donde según las decisiones que tome el usuario y los resultados que cosechen sus gladiadores al participar en distintos torneos y eventos, se van logrando superar los distintos retos que se le presentan al jugador.

El usuario abre la aplicación y tiene las distintas opciones de gestionar el juego a nivel de partida, como pueden ser continuar o empezar partida y el menú de opciones.

Si el usuario quiere continuar con la partida o comenzar una nueva, se introduce al menú de la gestión de la escuela. En él puede realizar todos aquellos cambios que considere necesarios para que el nivel de la escuela suba, como por ejemplo, compra/venta de gladiadores, entrenamientos, etc.

Las partidas son en modo offline, donde aparecen, según la semana en la que se esté, una serie de torneos que constan de uno o varios combates. El usuario solo puede guardar la partida al salir de la misma y no cuando quiera, ya que se premia la toma de decisiones correctas durante el transcurso del videojuego.

Todas estas funcionalidades se explican en el apartado *Funciones del producto* donde se definen con más detalle.

Para poder comprender la terminología de la aplicación, a continuación se muestran dos tablas con los conceptos y acrónimos utilizados. Se pretende que estas tablas sirvan de aclaración de algunos de los conceptos técnicos que serán utilizados a lo largo de este documento.

CONCEPTO	*DEFINICIÓN*
Hardware	Componente física de un sistema informático.
Software	Componente lógica de un sistema informático.
Sistema	Conjunto interrelacionado de componentes hardware, software y usuario.
Usuario/Jugador	Persona que interactúa con el sistema.

ACRÓNIMO	*SIGNIFICADO*
SW	Software
HW	Hardware
BD/BBDD	Base de datos/Bases de datos.
FIS	Sistema de Inferencia Fuzzy.
CPU	Unidad Central de Procesamiento.

3.2 Descripción general

La aplicación desarrollada consiste en un videojuego para el sistema operativo Android, donde aquellos usuarios a los que les guste el mundo del combate y de la gestión puedan potenciar sus habilidades. La aplicación está estructurada de tal forma que puede ir evolucionando a nivel arquitectónico y se pueden añadir nuevos requisitos software con facilidad, como jugar online con otros usuarios o aplicar la lógica fuzzy en otras áreas diferentes del proyecto.

La aplicación cuenta con una interfaz de usuario eminentemente gráfica. Es sencilla de manejar, por lo que los usuarios pueden empezar a jugar lo antes posible. Para conseguir este fin, la aplicación debe ser fluida y sobretodo, fácil de comprender, con la finalidad de que no decrezca la diversión del usuario respecto al juego.

A continuación se especifican los requisitos de usuario asignados a cada parte del sistema:

Menú Principal Aplicación: En este menú de la aplicación se agrupan las principales funcionalidades para gestionar el juego a nivel de partida completa y las opciones globales de la aplicación, como se describen a continuación:

- Nueva Partida

  Se muestra un mensaje informativo al usuario de que la partida anterior será borrada en caso de que quiera continuar. En caso de confirmación, se borra dicha partida; en caso contrario, puede seguir jugando a través de la opción de *Continuar Partida*.

- Continuar Partida

  Se procede a cargar los datos de la última partida jugada, que está almacenada en el propio dispositivo móvil, en una base de datos SQLite. En caso de que no existan datos previos guardados, se le notifica al usuario que debe comenzar una nueva partida.

- Opciones

  El usuario puede habilitar la opción de vibración en el combate, y tendrá la opción de cambiar la luminosidad de la pantalla del dispositivo móvil.

Menú Principal Usuario: En este menú se agrupan las principales funcionalidades que permiten al usuario gestionar los distintos aspectos de su partida.

- Gestión de la escuela
  - Consultar Escuela

    Permite al usuario ver todas las características de su escuela como dinero, fama y reputación. El usuario dispone de dos opciones donde puede consultar los empleados e instalaciones, o bien mejorarlas.

  - Mejorar Instalaciones

    El usuario ve en qué estado se encuentran sus instalaciones, con la opción de mejorar aquellas que lo necesiten. La mejora se hace mediante el pago de un cierto número de monedas, que varía según el nivel en el que se encuentre la escuela.

  - Mejorar Empleados

    Esta opción posee las mismas funcionalidades que la anterior, pero en este caso se mejoran los empleados.

  - Espiar Escuela Rival

    El usuario, bajo pago de 500 monedas, puede ver el estado de la escuela rival y sus gladiadores, pudiendo así poder tomar decisiones más acertadas.

- Gestión de los gladiadores:
    - Consultar Gladiadores

      Esta opción muestra las características de los gladiadores.

    - Comprar Gladiadores

      El usuario puede ampliar el número de gladiadores de su escuela accediendo al mercado de gladiadores. Allí puede pujar por los gladiadores cuyas características le convengan más, siempre y cuando tenga monedas suficientes.

    - Equipar Gladiadores

      El usuario puede equipar a sus gladiadores con las armas que tenga en la escuela.

    - Entrenar Gladiadores

      El usuario puede definir dos tipos de entrenamiento. Uno de ellos, para mejorar aquellas características consideradas primarias, como fuerza y agilidad. El otro tipo mejora los estilos de combate del gladiador, como el manejo de armas con ambas manos o la habilidad cuerpo a cuerpo. Al avanzar la semana, en dichas características se puede apreciar una mejoría.

- Gestión de la partida
    - Avanzar Semana

      Funcionalidad de vital importancia, ya que de ésta depende que se notifiquen los eventos, que estarán guardados en la base de datos de manera prefijada. Avanzamos en el tiempo, lo que se traduce en diferentes tipos de cambios.

    - Salir

      Se guarda el estado de la escuela en ese mismo instante, con todos los cambios que haya realizado el usuario hasta ese momento. Al tratarse de un juego en el que tienen mucho peso las decisiones que se tomen, la opción de no guardar partida y volver a un instante anterior de la misma no es posible.

    - Notificaciones de Combate

      Los eventos en los que participa el usuario están almacenados en la base de datos del juego de forma predeterminada. Para un evento se define su recompensa económica, el tipo de combate, su número de

combates y una descripción del torneo. A medida que se avance en la semana de juego, se notificarán al usuario en la barra de estado del dispositivo móvil los eventos programados para dicha semana.

- Combate

  El usuario puede elegir cualquier gladiador que posea en su escuela y enfrentarlo al gladiador de la CPU. Se proclama un ganador, mediante un algoritmo que se explica en la sección *4.2- Algoritmos*, y una vez acabado el combate, se muestra un resumen con animaciones 2D donde se puede ver como se ha desarrollado el combate.

El público objetivo de este videojuego son usuarios que quieran potenciar su capacidad de gestión y sentir la responsabilidad e impacto que conlleva la toma de decisiones y su ejecución en el juego. Esto, por supuesto, no quiere decir que el juego esté orientado únicamente a este tipo de usuario. Una vez la aplicación se suba al Android Market, se espera que interese a jugadores ocasionales y habituales, incluso a aquellos reacios a este formato de juegos, gracias al aspecto sencillo y limpio de las interfaces del juego, a la no necesidad de conexión a Internet para jugar, y sobre todo a su escaso tamaño que hace que pueda descargarse en la inmensa mayoría de dispositivos móviles actuales.

Se ha estimado un tiempo de desarrollo del proyecto de ocho meses a partir de la fecha de inicio (octubre 2013). Debido al límite de tiempo impuesto por el curso académico, no podremos realizar todas las tareas deseables. Pero si se proponen en el apartado de *Requisitos Futuros.*

Las soluciones planteadas a los requisitos especificados en este documento requieren unas mínimas características hardware y software de los dispositivos móviles en los que se puede instalar la aplicación, para permitir su correcto funcionamiento. El sistema requiere:

- Sistema operativo Android.
- Dispositivo móvil.
- APIS objetivo y mínima de Android 2.2 (Gingerbread). Compatibilidad al menos hasta la versión 4.1 (Jelly Bean).
- Ningún requisito hardware en especial.

Como trabajo futuro se prevé que la aplicación pueda conectarse a un servidor de bases de datos para permitir jugar online a distintos jugadores. El primer paso en esta línea sería que los jugadores pudieran enfrentar a sus luchadores. Con esta modificación, el juego pasaría a tener un modo multi-jugador. En esta modalidad se podrá implementar un ranking de aquellos usuarios que tengan la máxima puntuación de entre todos los jugadores que se conecten de manera online.

Otra posible ampliación para el modo multi-jugador sería un registro explícito en el mismo. En este registro se enviaría al usuario un correo de confirmación antes de

habilitarle para este modo.

En el futuro también se plantea que el jugador tenga la opción de "mudarse" a otra ciudad. Esto le será posible cuando haya alcanzado un cierto nivel en el juego. La mudanza consistirá en vender la escuela y comprar otra en una ciudad mejor. El jugador podrá llevarse a su nueva escuela sus gladiadores, empleados y equipamiento.

3.3 Requisitos específicos

A continuación se muestra el detalle de requisitos de sistema que cumple nuestro entorno. Para ello se ha optado por una especificación estructural basada en tablas descriptivas de cada actividad, en cada menú de la aplicación.

Menú Principal Aplicación

Nombre	Nueva Partida.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Inicio de una nueva partida en el juego.
Entrada	Nada.
Salida	Mensaje informativo.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso al sistema para borrar la base de datos existente (en caso de que la hubiera) y creación de una nueva.
Acción	Se crea una nueva base de datos.
Precondición	Ninguna.
Postcondición	Inicio de una nueva partida.
Efectos Laterales	Se borrará cualquier información previamente almacenada.

Nombre	Continuar Partida.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Se continúa con una partida ya existente.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso a la base de datos.
Acción	El usuario pulsa el botón y automáticamente se carga la partida en el estado previo en donde el jugador la dejó.
Precondición	Hay una partida correctamente almacenada.
Postcondición	Se continúa con la partida.
Efectos Laterales	Ninguno.

Nombre	Opciones.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Se accede el menú de opciones.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del usuario.
Necesita	Nada.
Acción	Se carga el menú de opciones donde podemos controlar tanto la vibración como el brillo.
Precondición	Ninguna.
Postcondición	Ninguna.
Efectos Laterales	Ninguno.

Menú Principal Jugador

Nombre	Consultar Escuela.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Muestra información acerca de la Escuela del usuario como el nombre, dinero, nivel, número de habitaciones o reputación.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema/Interfaz del usuario.
Necesita	Acceso a la base de datos.
Acción	Se accede a la base de datos y se muestran los distintos campos de información al usuario.
Precondición	Que haya una partida en juego.
Postcondición	Ninguna.
Efectos Laterales	Ninguno.

Nombre	Mejorar Empleados.
Prioridad	Media.
Estabilidad	Alta.
Descripción	Muestra información acerca de los empleados de la escuela, otorgando la posibilidad de mejorar las habilidades de cada uno.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema/Interfaz del usuario.
Necesita	Acceso a la base de datos.
Acción	Se accede a la base de datos y se muestran los empleados, con su nivel correspondiente.
Precondición	Que haya una partida en juego.
Postcondición	Ninguna.
Efectos Laterales	Ninguno.

Nombre	Mejorar instalaciones.
Prioridad	Media.
Estabilidad	Alta.
Descripción	Muestra información acerca de las instalaciones de entrenamiento, otorgando la posibilidad de mejorar el nivel de cada uno.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema/Interfaz del usuario.
Necesita	Acceso a la base de datos.
Acción	Se accede a la base de datos y se muestran las instalaciones, con su nivel correspondiente, el cual es mejorable siempre que se cumplan ciertas condiciones.
Precondición	Que haya una partida en juego.
Postcondición	Ninguna.
Efectos Laterales	Ninguno.

Nombre	Espiar Escuela Rival.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Observar el estado de la escuela de la máquina rival.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz de Usuario.
Destino	Interfaz de Usuario.
Necesita	Acceder a la estructura de datos que contiene la información de las escuelas.
Acción	El usuario puede ver las características de los gladiadores y de la escuela pagando una cantidad fija de monedas.
Precondición	Partida comenzada.
Postcondición	Características de la escuela rival.
Efectos Laterales	Ninguno.

Nombre	Consultar Gladiadores.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Se muestran los gladiadores que posee el usuario, pudiéndose consultar sus características y equiparles de cara al combate.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del usuario/Interfaz del sistema.
Necesita	Acceso a la base de datos.
Acción	Se cargan los gladiadores pertenecientes al usuario y se permite además, equiparlos con las armas que posea en su escuela, es decir, puede ver la configuración de un gladiador una vez armado.
Precondición	Gladiadores del usuario almacenados en la base de datos.
Postcondición	Se carga la interfaz de Gladiadores.
Efectos Laterales	Ninguno.

Nombre	Comprar Armas.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Permite adquirir armas.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Una partida en juego.
Acción	Al comprar el arma, podrá utilizarla para equipar a cualquier gladiador que tenga en su escuela.
Precondición	Ninguna.
Postcondición	Arma adquirida.
Efectos Laterales	El arma no volverá a estar en el mercado.

Nombre	Vender Gladiadores.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Vender un gladiador de la escuela.
Entrada	Gladiador.
Salida	Precio Gladiador.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Gladiador en la escuela.
Acción	Se vende un gladiador de la escuela, aumentado el número de monedas del usuario.
Precondición	Gladiador.
Postcondición	Gladiador vendido y aumento de dinero.
Efectos Laterales	El gladiador desaparece de la escuela.

Nombre	Comprar Gladiadores.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Permite la compra de un gladiador.
Entrada	El nombre del gladiador.
Salida	Un booleano.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso a la base de datos.
Acción	Se elimina el gladiador del mercado y se introduce como nuevo gladiador del usuario, modificando la base de datos.
Precondición	Gladiadores disponibles en el mercado.
Postcondición	La compra de un gladiador.
Efectos Laterales	Ninguno.

Nombre	Entrenar Gladiadores.
Prioridad	Media.
Estabilidad	Alta.
Descripción	Se carga la interfaz que permite el entrenamiento de los gladiadores comprados por el usuario.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso a la base de datos.
Acción	El usuario tiene dos opciones de entrenamiento, una primera que mejora aquellas habilidades que son prioritarias y la otra opción que mejora las demás, todo ello con la ayuda de la opción de características para que el usuario sepa que mejora en todo momento.
Precondición	Que el usuario posea algún gladiador.
Postcondición	Modificación de la base de datos.
Efectos Laterales	Mejora de las habilidades de los gladiadores.

Nombre	Características Gladiadores.
Prioridad	Media.
Estabilidad	Alta.
Descripción	Información de un gladiador.
Entrada	Gladiador.
Salida	Características del gladiador.
Origen	Interfaz del usuario.
Destino	Interfaz del usuario.
Necesita	Acceder a la estructura de datos que contiene a los gladiadores.
Acción	Selección de un gladiador y se mostraran las características del mismo.
Precondición	Que el usuario posea algún gladiador.
Postcondición	Características del gladiador.
Efectos Laterales	Ninguno.

Nombre	Equipar Gladiadores.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Equipar con armamento un gladiador.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema, usuario.
Necesita	Acceso a la estructura de datos de armas.
Acción	Se equipa al gladiador con aquellas armas de que se disponga en la escuela.
Precondición	Que el usuario posea algún gladiador y armas.
Postcondición	Modificación de su armamento.
Efectos Laterales	Gladiador con equipamiento asignado.

Nombre	Avanzar Semana.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Permite al usuario avanzar una semana en el juego, produciéndose diversas acciones simultaneas tras su ejecución.
Entrada	Nada.
Salida	Un booleano.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso a la base de datos, tanto como para su lectura como escritura.
Acción	Al avanzar una semana, transcurren diferentes cambios, como la mejora de las habilidades de los gladiadores, curaciones para los que han sido lesionados en combate, etc. También cabe la posibilidad de que se genere un nuevo evento.
Precondición	No tiene que haber ningún evento sin explorar en la barra de notificaciones, debido a que habría una inconsistencia entre la semana actual y la semana en la que se celebra el combate.
Postcondición	Se generan diferentes cambios en la base de datos.
Efectos Laterales	Modificaciones en las características de los gladiadores y eventos. La escuela rival realiza su gestión de instalaciones, mercado y equipación de los gladiadores.

Nombre	Guardar Partida.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Almacenar partida en curso.
Entrada	Todos los cambios que se han realizado durante la partida.
Salida	Nada.
Origen	Interfaz de usuario.
Destino	Interfaz de sistema.
Necesita	Acceso a la base de datos.
Acción	Se guardan todo los cambios que ha realizado el usuario durante la partida, como puede ser la compra/venta de gladiadores, la equipación de cada uno de ellos, etc. Se guarda partida en Salir.
Precondición	Que haya una partida en juego.
Postcondición	Partida guardada.
Efectos Laterales	Ninguno.

Nombre	Salir.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Se finaliza momentáneamente la partida actual.
Entrada	Nada.
Salida	Nada.
Origen	Interfaz del usuario.
Destino	Interfaz del sistema.
Necesita	Acceso a la base de datos.
Acción	Se cierra la partida actual, guardándose el estado de esta. No se dará opción de no guardar al tratarse de un juego de toma de decisiones.
Precondición	Que haya una partida en juego.
Postcondición	Se cierra la partida actual.
Efectos Laterales	Ninguno.

Nombre	Notificaciones.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	Aviso de combates al usuario.
Entrada	Fecha del combate.
Salida	Generación del evento.
Origen	Interfaz del sistema.
Destino	Interfaz de usuario.
Necesita	Acceso a la estructura de datos que contiene los eventos.
Acción	El sistema comprueba si para la semana en la que el usuario se adentra, hay un evento para participar. Los eventos están almacenados en la base de datos de manera predeterminada. Aparecerá un icono con la forma del logo del juego en la barra de notificaciones del dispositivo móvil, donde el usuario tiene que decidir si participar o no.
Precondición	Que haya una partida en juego.
Postcondición	Notificación generada.
Efectos Laterales	Ninguno.

Nombre	Combate.
Prioridad	Alta.
Estabilidad	Alta.
Descripción	El usuario se enfrenta contra la máquina.
Entrada	Dos gladiadores que se van a enfrentar.
Salida	Animación en 2D.
Origen	Interfaz de usuario.
Destino	Interfaz de usuario.
Necesita	Acceso a la información de la escuela del usuario.
Acción	El jugador enfrenta a su luchador contra otro de la máquina donde se mostrará una animación gráfica en 2D y un resumen de la recompensa.
Precondición	Que el usuario tenga luchadores.
Postcondición	Las características de dichos luchadores modificadas.
Efectos Laterales	Que las características del luchador de la máquina se modifiquen.

3.4 Restricciones de diseño

El mercado sobre el que operará nuestra aplicación es, con mucha frecuencia, un entorno frenético donde cada día que pasa hay nuevas aplicaciones de diferentes temáticas y gráficamente mejor desarrolladas. Por este motivo cabe destacar una restricción que será tomada como una máxima en el desarrollo de las interfaces de usuario:

El entorno debe ser sencillo, amigable, fluido y muy visual. Un ejemplo de este tipo de aplicaciones lo podemos ver en el juego *POU*[4], tamagochi de la era Android, donde la interfaz es muy intuitiva y al usuario no le cuesta mucho tiempo dominar el juego. Otros ejemplos, como Angry Birds, demuestran que una de las claves del éxito en el mundo de los videojuegos radica en la facilidad de comprender intuitivamente y sin necesidad de extensas instrucciones el funcionamiento de dicho videojuego.

El proyecto se ha desarrollado en el entorno Eclipse y se ha divido en paquetes, donde cada uno de ellos realiza una función diferente. Esta estructuración permite trabajar de una manera mucho más sencilla, siendo posible modificar o acceder a cualquier estructura del código sin problemas.

Se ha considerado como mejor opción la codificación a través de un lenguaje orientado a objetos, en nuestro caso, Java.

3.5 Atributos del sistema software

La arquitectura de nuestro proyecto y las características de su desarrollo se resumen a través de las siguientes características que posee:

- Aplicación de gestión.
- Persistencia de datos a través de una base de datos relacional.
- Desarrollo incremental y evolutivo.

3.6 Software y Herramientas Utilizadas

Aquí se exponen los distintos programas que se han usado en el desarrollo del proyecto:

- Adobe PhotoShop: se trata esencialmente de una aplicación informática que está destinada a la edición, retoque fotográfico y dibujo a base de imágenes de mapa de bits.[5].
- Eclipse: plataforma de desarrollo de código abierto [8] basada en Java. Es un conjunto de servicios para construir un entorno de desarrollo a partir de plugins.[6].
- MatLab: herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio(lenguaje M) [7].
- GIMP: programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías [9].

El proyecto ha sido realizado íntegramente en Android. Este es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tabletas. Entre sus características principales podemos encontrar:

- Biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de OpenGL ES 2.0.

- Almacenamiento en SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos. La principal ventaja de este sistema de gestión de bases de datos consiste en su capacidad de enlazarse con el programa pasando a ser parte integral del mismo. El programa puede usar SQLite a través de llamadas simples a funciones, y todo el conjunto que forma la base de datos puede ser alojado dentro del dispositivo como si se tratase de un solo fichero de tipo estándar. Debido a su escaso tamaño, se recomienda su uso para sistemas integrados como Android [10].

- Bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java.

Dentro de una aplicación Android, podemos destacar los siguientes componentes básicos [11]:

- *Activity*: Una actividad o *activity* es el componente básico que permite crear componentes que interaccionan con el usuario. Cada actividad tiene asociada una vista o *view*. La vista constituye el conjunto de elementos gráficos, que se presentan al usuario para que interaccione con el sistema.

- *Intent*: Una intención o *intent* es un componente básico de Android, que permite el envío de mensajes entre componentes. El paso de mensajes a través de *intents* puede considerarse una facilidad para asociar (binding) componentes de forma tardía y en tiempo de ejecución entre la misma aplicación o diferentes aplicaciones.

- *Service*: Un componente de la aplicación que permite ejecutar operaciones largas en segundo plano y que no facilita una interfaz de usuario.

- *Content providers*: Manejan el acceso a información estructurada. Encapsulan la información y proporcionan mecanismos para facilitar la seguridad.

- *Processes and threads*: Cuando una aplicación se ejecuta, Android ejecuta un proceso con un solo hilo de ejecución. Por defecto todas las aplicaciones son ejecutadas bajo el mismo proceso e hilo.

- *Android Manifest*: Todas las aplicaciones han de contener el archivo AndroidManifest.xml en su directorio raíz. Este archivo, contiene información esencial para el sistema operativo sobre la aplicación.

- *User interface*: La interfaz del usuario contiene todo con lo que el usuario puede interactuar o ver. Android facilita una serie de componentes, como por ejemplo objetos de estructuración para *layouts* o controles para la interfaz de usuario [12].

3.7 Hardware Utilizado

En este apartado, se presentarán los distintos dispositivos móviles donde se ha instalado la aplicación Ludus Manager, una vez que ya estaba lista para poder jugar, y así buscar fallos que tuviese la aplicación.

Cuando se comenzó la aplicación, las pruebas se realizaban en el emulador de Android, como el que se muestra en la Fig.4.

Fig.4. Emulador de Android

A continuación se detallan todos los dispositivos móviles en los que ha sido testeada la aplicación, con sus especificaciones completas y las versiones del sistema operativo Android usadas:

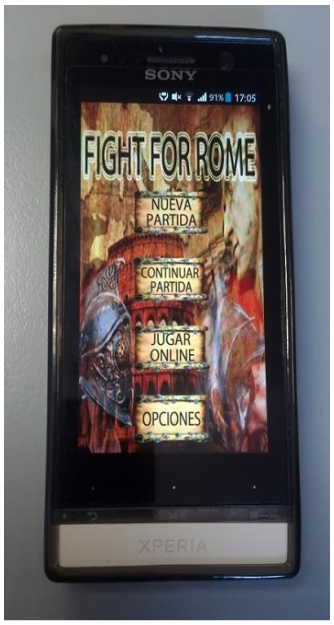
Modelo	Orange Monte Carlo
Versiones Android	2.3 (Gingerbread)
Dimensiones	126 x 67 x 11.3 mm
Pantalla	480 x 800, 4.3 pulgadas
CPU	800 MHz
RAM	512MB
Batería	1400 mAh

Orange Monte Carlo

Modelo	Motorola RAZR
Versiones Android	4.0.4 (Ice Cream Sandwich) y 4.1.2 (Jelly Bean)
Dimensiones	130.7 x 68.9 x 7.1 mm
Pantalla	540 x 960, 4.3 pulgadas
CPU	Dual-Core 1.2 GHz
RAM	1GB
Batería	1780 mAh

Motorola RAZR

Modelo	Sony Xperia U
Versiones Android	2.3 (Gingerbread) y 4.0.4 (Ice Cream Sandwich)
Dimensiones	112 x 54 x 12 mm
Pantalla	480 x 854, 3.5 pulgadas
CPU	Dual-Core 1 GHz
RAM	512MB
Batería	1320 mAh



Sony Xperia U

CAPITULO 4. DESARROLLO DE LA APLICACIÓN

4.1 Decisiones en el proyecto

Una de las primeras decisiones tomadas a la hora de empezar a desarrollar el videojuego, fue la de crear una interfaz sencilla y fácil de manejar para que el usuario pueda empezar a jugar de manera rápida.

Este proyecto cuenta con características peculiares dentro del sistema, como la existencia de un sistema de inferencia fuzzy (FIS), que calcula el valor de un gladiador de la escuela mediante el método de inferencia de Mamdani [13]. Como ya hemos mencionado anteriormente, dicha técnica de inteligencia artificial se podrá aplicar a futuros requisitos del proyecto.

A nivel de programación en Android, se ha estructurado el proyecto en diferentes paquetes que se distinguen por su funcionalidad: lógica fuzzy, activities, Javacode, inteligencia artificial del rival y gestión de animaciones mediante sprites. De ésta manera se ha logrado mantener siempre la división de trabajo entre cada uno de los pilares del proyecto, permitiendo un trabajo simultáneo por parte de los miembros del proyecto.

En cuanto a las activities, el principal uso que hicimos de los recursos gráficos que ofrece Android son los botones, Gallery, Canvas y eventos para los menús, y animaciones para recrear los combates entre gladiadores.

4.2 Diseño

Casos de uso.

A partir de los requisitos expuestos en el capítulo anterior se han identificado los casos de uso correspondientes a la aplicación. A continuación se van a detallar mediante diagramas.

- Menú Principal Aplicación

El usuario tendrá tres opciones cuando arranque la aplicación en su dispositivo móvil. Continuar la partida que ya tiene empezada, empezar una nueva o configurar los ajustes de luminosidad o vibración del juego, según muestra la Fig.5.

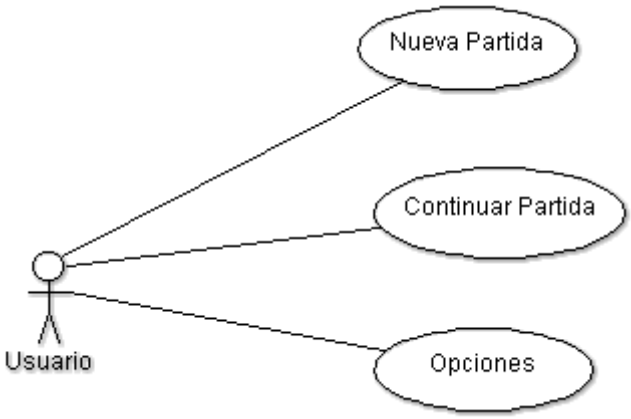


Fig. 5. Diagrama de Casos de Uso de Menú Principal Aplicación

- Menú Principal Jugador

Debido al gran número de requisitos que tiene la parte del menú principal del usuario, se dividirán en subdiagramas para que sea más sencillo, según se muestra en la Fig.6.

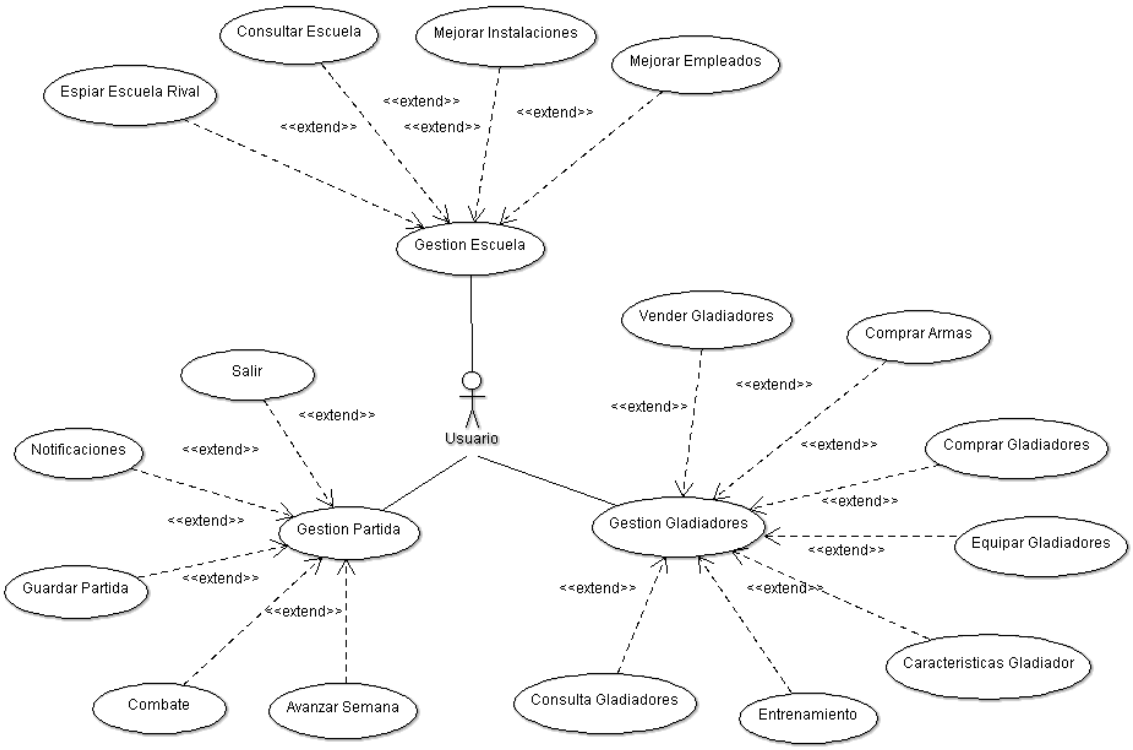


Fig. 6. Diagrama de Casos de Uso Menú Principal

4.3 Paquetes y clases

El proyecto consta de los paquetes descritos a continuación, como puede verse en la Fig.7.

- Inteligencia Artificial.

  Paquete que contiene las clases que se encargan de gestionar las características de la escuela rival. Dichas clases se encargarán de construir un conjunto de posibles acciones que puede hacer la máquina según sus características, para así elegir la más acertada en cada momento.

- Lógica Fuzzy.

  Clases que implementan el sistema de inferencia fuzzy. Dicho sistema, que se explicará con más detalle en el capítulo 5 de *Técnicas Especiales Utilizadas*, se encargará de calcular el precio de venta de un gladiador dependiendo de ciertas características del mismo.

- Activities.

  Este paquete está formado por todas las interfaces gráficas del juego, como MenuPrincipal, Combate, etc.

- Algoritmos.

  Todo lo que se refiere a creación, inicialización y gestión de gladiadores, armamento, mercado, ludus, ciudad y eventos es de lo que se encarga este conjunto de clases. La inicialización y gestión de la base de datos SQLite también está contenida en éste paquete.

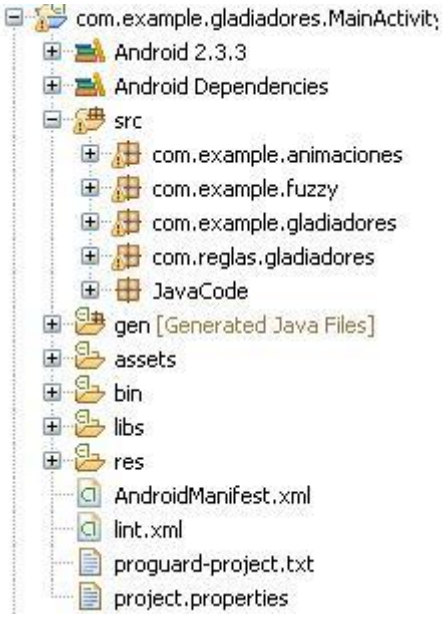


Fig. 7. Paquetes del proyecto.

Inteligencia Artificial

Paquete que se encarga de la gestión de las mejoras que tiene que realizar la escuela rival, donde analizando sus instalaciones de fuerza, destreza, agilidad y resistencia, se le indica a la máquina que prioridad de mejora tiene cada una de ellas. El conjunto de reglas se explican de forma resumida, a continuación:

- *NivelEntrenador*, donde tiene más prioridad que las instalaciones, ya que sin un buen entrenador, los gladiadores no mejorarán de la misma manera que si tiene un buen nivel el entrenador, independientemente de lo buenas que sean las instalaciones que tenga la máquina.
- *Mejora de las instalaciones*: se compara el nivel de una instalación con el resto. El procedimiento es el siguiente:
    - Si el nivel de una instalación en concreto, es muy inferior a las demás, tiene prioridad alta.
    - Si el nivel de una instalación es igual a las demás, tiene prioridad media.
    - Si el nivel de la instalación es mayor que las demás, su prioridad es baja.

Las distintas clases Java que implementan la inteligencia artificial de la escuela rival están contenidas en el paquete *com.reglas.gladiadores*, como puede verse en la Fig.8.

Fig. 8. Clases del paquete Reglas

- *Acciones.java*

  Clase que genera las acciones, una vez son tomadas por *EleccionCriterio.java* y *ReglasLudus.java*, teniendo como atributos una descripción textual de la mejora a realizar por la escuela rival y un valor numérico que representa la importancia de dicha mejora.

- *EleccionCriterio.java*

  Esta clase recibe una lista con todas las acciones que se puedan tomar. Se ejecutan las más prioritarias mediante el método *EjecutarReglas*, y se gestiona el ludus mediante la clase *Ludus.java* que se verá más adelante.

ReglasLudus.java

Clase que contiene el conjunto de reglas mediante las cuáles se regirá el comportamiento de la inteligencia artificial del rival. Posee métodos para analizar el estado actual del ludus rival, y según los resultados, asigna diferentes prioridades a cada una de las posibles acciones.

La organización de estas tres clases se puede apreciar con más detalle en la Fig.9.

Fig. 9. Diagrama de clases del paquete Reglas.

Lógica Fuzzy

Las distintas clases Java que implementan las *activities* están contenidas en el paquete *com.example.fuzzy*, como puede verse en la Fig.10.

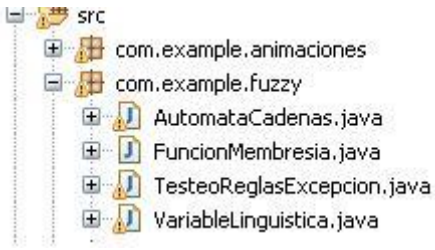


Fig. 10. Clases del paquete Fuzzy

- *AutomataCadenas.java*

  Clase que se encarga de procesar las reglas difusas definidas en la clase *ActivityVender*, mediante un autómata.

- *FuncionMembresia.java*

  Clase que guarda el nombre y valor del conjunto difuso que esté asociado a una variable lingüística. Dichas variables lingüísticas se definen en la clase *ActivityVender.*

- *TesteoReglasExcepcion.java*

  Clase que hereda de *Exception*, cuya finalidad es comprobar si hay algún error sintáctico cuando analiza la regla.

- *VariablesLinguistica.java*

  Clase que contiene el método de defuzzificar mediante el método del Centroide que será llamado por la variable de salida. También contiene los métodos que definen los mínimos de cada variable lingüística de salida.

La organización de estas clases se aprecia con más detalle en la Fig.11.

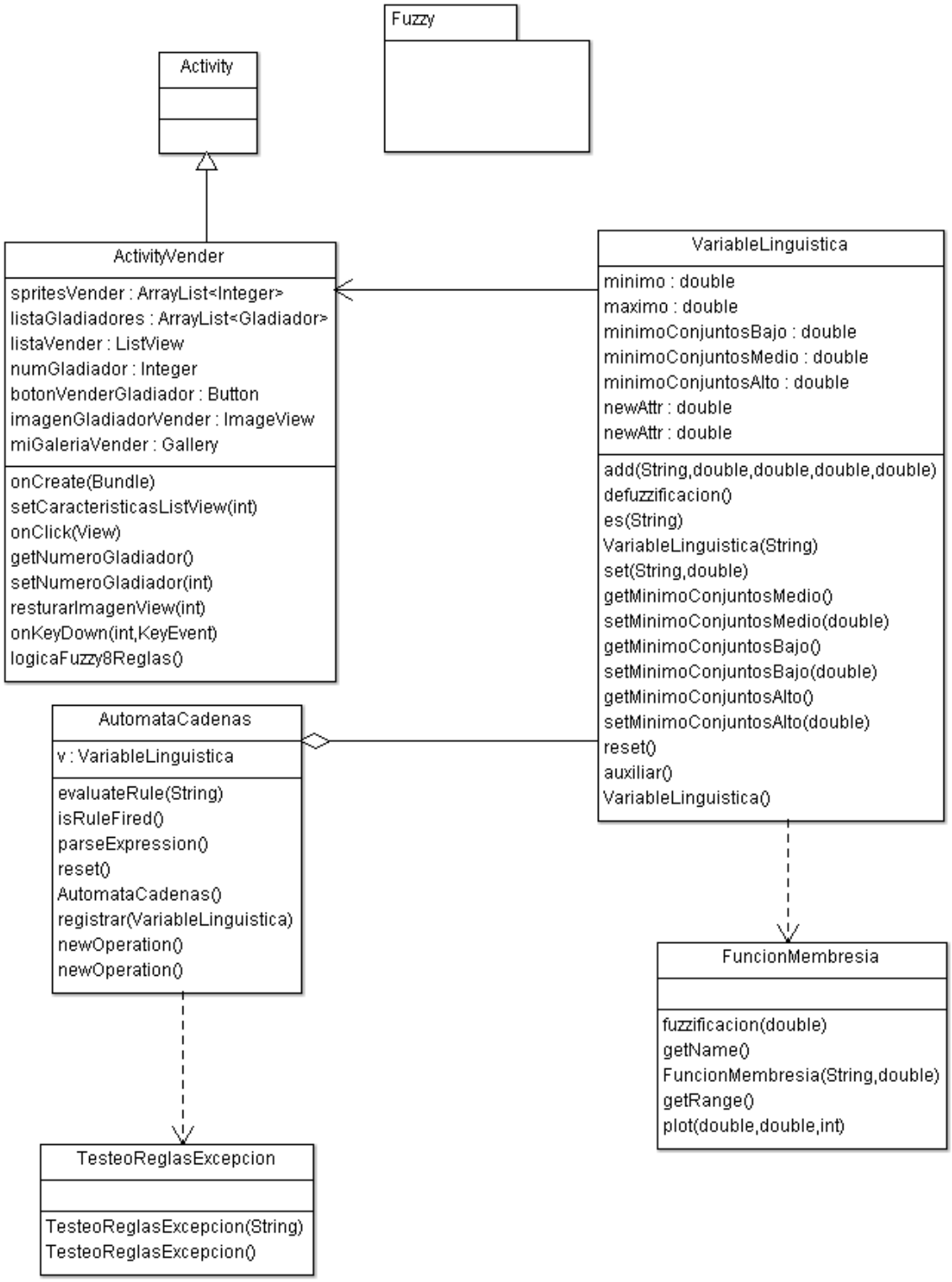


Fig. 11. Diagrama de clases del paquete Fuzzy

Activities

Las distintas clases java que implementan las activities están contenidas en el paquete *com.example.gladiadores*, como puede verse en la Fig.12.

Fig. 12. Clases del paquete *Activities*

A continuación se presenta el diagrama de clases del paquete *com.example.gladiadores*, donde están las clases que heredan de la clase *Activity*. En esta memoria solo se han representado aquellas de mayor importancia, pero pueden consultarse en el CDROM adjunto al trabajo.

- *ActivityCaracteristicas.java*

  Muestra las características de cada gladiador.

- *ActivityCombate.java*

  En esta clase se permite equipar a los gladiadores, mediante cuadros de selección. Al pulsar el botón de participar, comienza el combate (animaciones) y se calcula el vencedor mediante las reglas de combate. Su diagrama de clases se puede ver en la Fig.13.

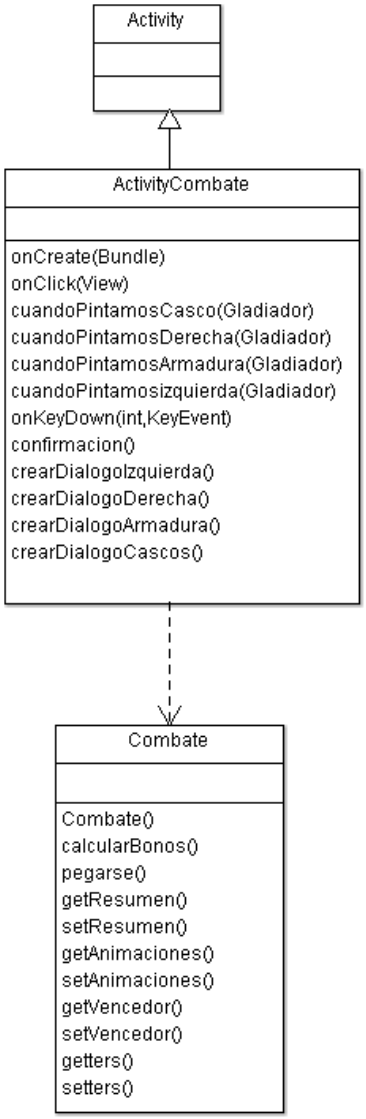


Fig. 13. Diagrama de clases de Combate

- *ActivityConsultaEscuela.java*

  Esta clase nos permite acceder a las mejoras de las instalaciones o los empleados. También muestra la información de nuestra escuela, como dinero, habitaciones, gladiadores, nivel o fama. Su diagrama de clases puede verse en la Fig.14.

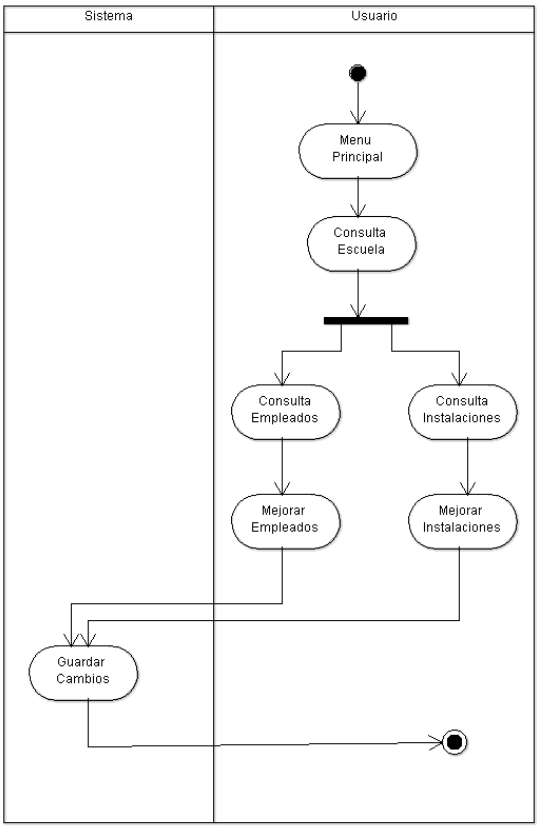


Fig. 14. Diagrama de Consulta Escuela

- *ActivityEmpleados.java*

  Desde este menú se puede mejorar el nivel de los empleados del Ludus. Se usa un *item adapter* con un *arraylist* de empleados que se muestra en forma de lista, y permite, pulsando un botón, mejorar de forma independiente el nivel de cada uno.

- *ActivityEntrenamiento.java*

  Desde este menú podemos asociar, de forma gráfica, un entrenamiento a cada uno de los gladiadores disponibles en nuestro Ludus. Mediante un *ImageGallery* se puede seleccionar un gladiador determinado, y asociarle cualquier entrenamiento a través de los diálogos emergentes. Su diagrama de clases puede verse en la Fig.15.

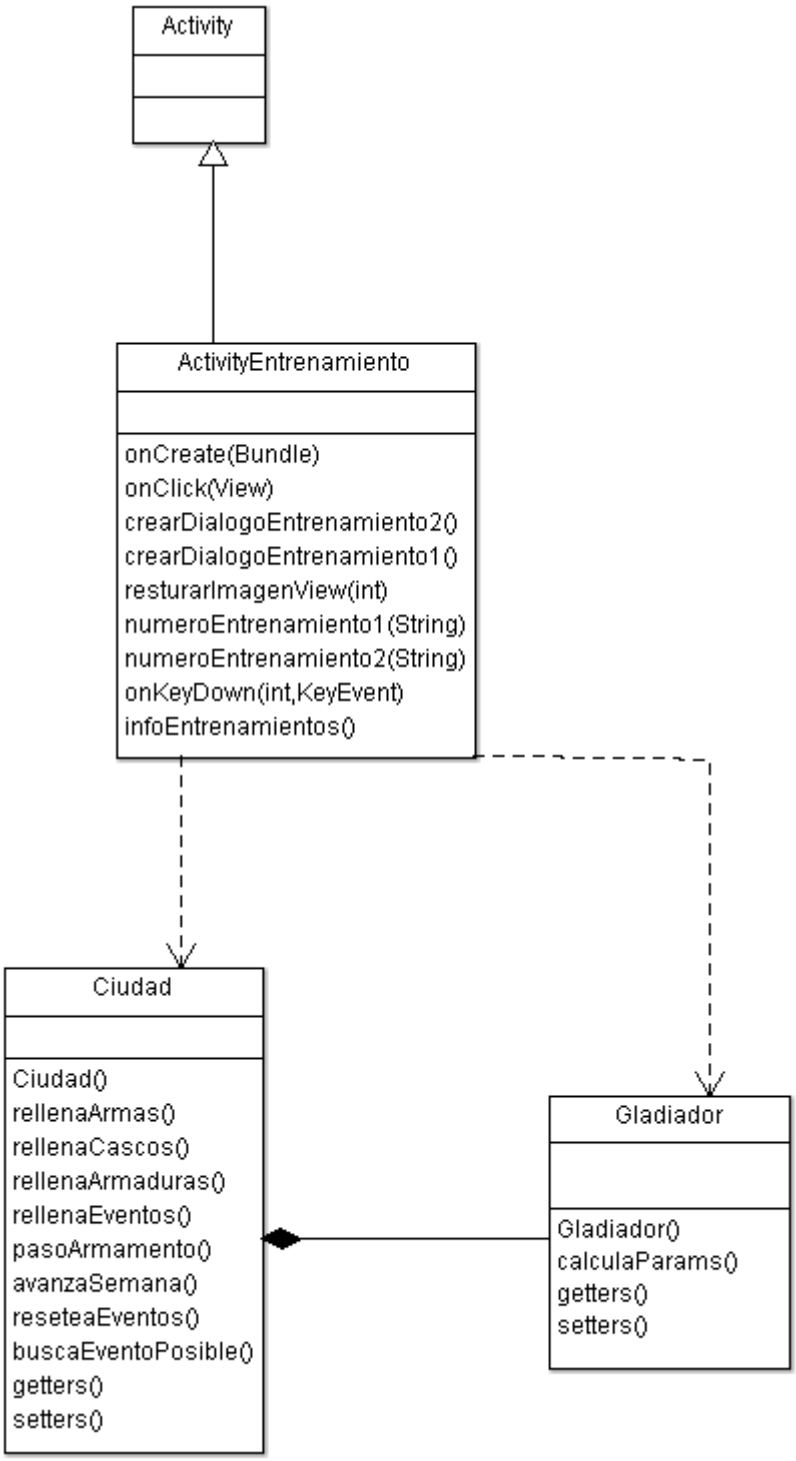


Fig. 15. Diagrama de clase de Entrenamiento

- *ActivityEspiarEscuela.java*

  La funcionalidad de esta actividad, es observar las características de la escuela rival, bajo previo pago de una cantidad fija de monedas.

- *ActivityespiarGladiadores.java*

  Interfaz donde se le muestra al usuario cada uno de los gladiadores de la escuela rival y sus características.

- *ActivityEvento.java*

  Nos muestra información sobre el evento en el que vamos a participar. Los eventos tienen información sobre la recompensa en caso de ganar el combate, el número de combates que se producirán y una breve descripción. Los eventos se cargarán de la base de datos, donde estarán ya definidos.

- *ActivityGanador.java*

  Información acerca de la recompensa que ha obtenido el gladiador del usuario tras ganar un combate. Se le mostrará el incremento de la fama y del dinero.

- *ActivityGladiadores.java*

  En esta clase se puede equipar a nuestros gladiadores con las armas que se hayan comprado previamente en el mercado. También se pueden consultar todas las características del gladiador como su fuerza, su destreza, etc. Su diagrama de clases puede verse con detalle en la Fig.16.

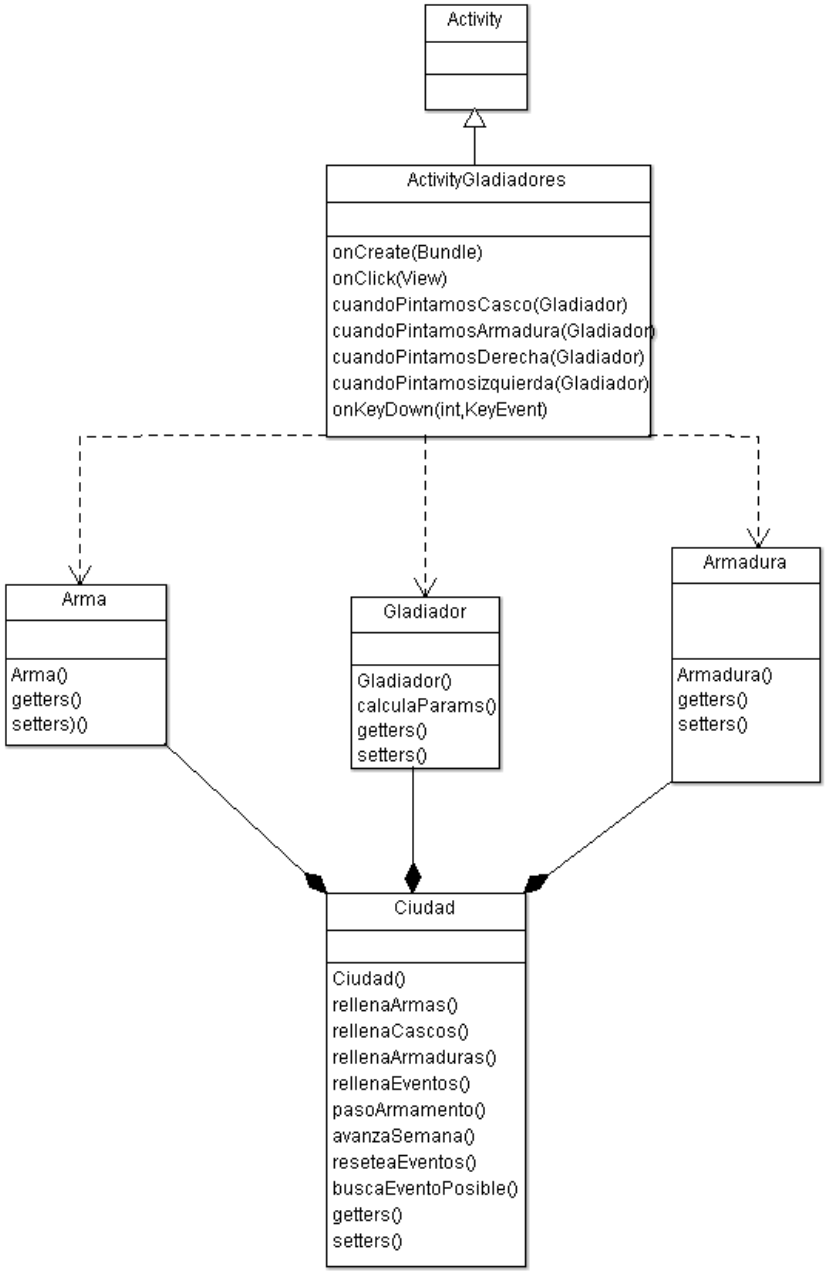


Fig. 16. Diagrama de clases Consulta de Gladiadores

- *ActivityInstalaciones.java*

  Clase que muestra el valor que tienen las instalaciones de la escuela del usuario.

- *ActivityMenuPrincipal.java*

  Clase que contiene el menú principal de la aplicación. Desde aquí se puede acceder a todas las opciones para la gestión del Ludus. En caso de que al avanzar una semana se produzca un evento, éste se verá desplegado en la barra de notificaciones del móvil.

- *ActivityMercado.java*
  En esta clase se representa el acceso al mercado de gladiadores. El jugador puede comprar un gladiador siempre que tenga suficiente dinero y espacio en su ludus. Tras la compra, el gladiador quedará almacenado en el ludus y aparecerá automáticamente en *ActivityGladiador* y en *Entrenamiento*. Su diagrama de clases se puede ver en la Fig.17.

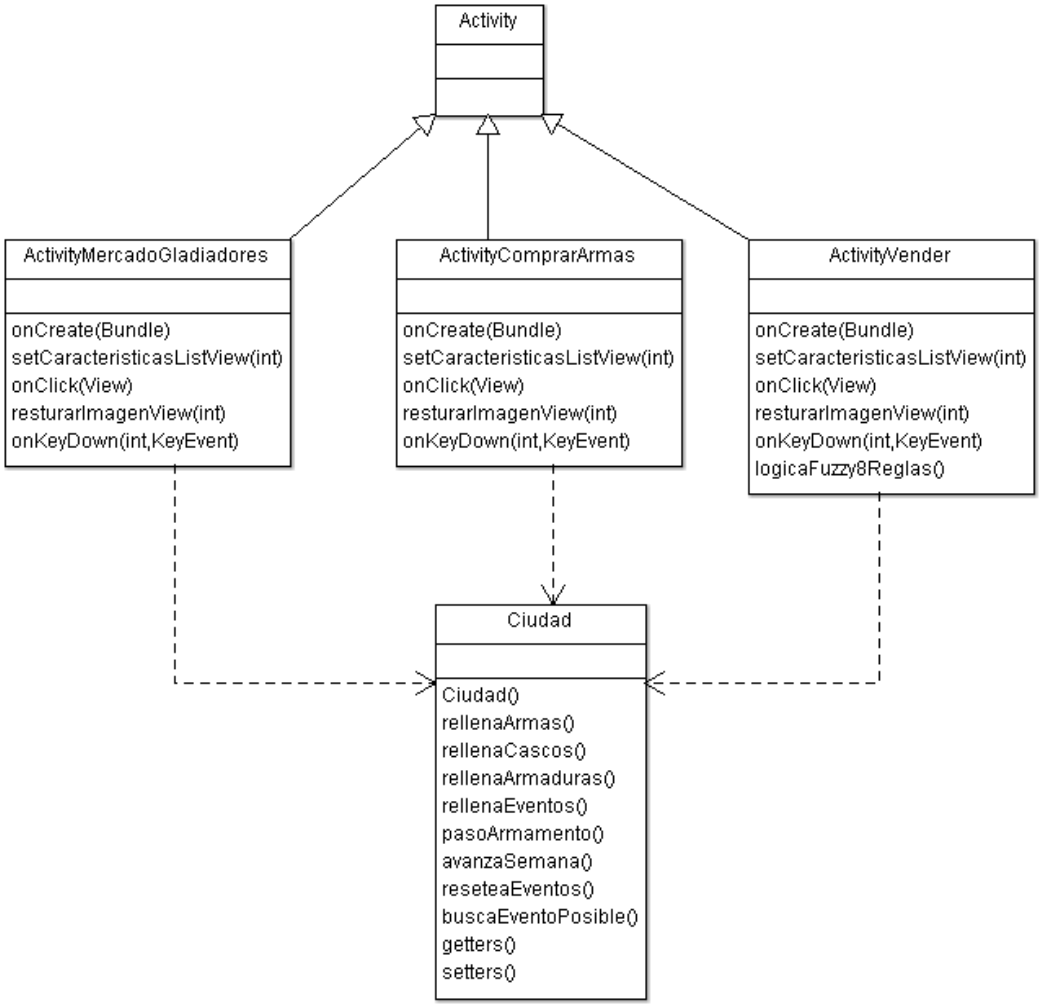


Fig. 17. Diagrama de clases de Mercado

- *ActivityMercadoArmas.java*

  Aquí el usuario puede comprar armas para equipar a sus gladiadores para el combate.

- *ActivityNuevaPartida.java*

  Desde esta clase se puede iniciar una nueva partida, borrando el contenido de la base de datos que hubiera previamente almacenado y generando una nueva. Su diagrama de clases está detallado en la Fig.18.

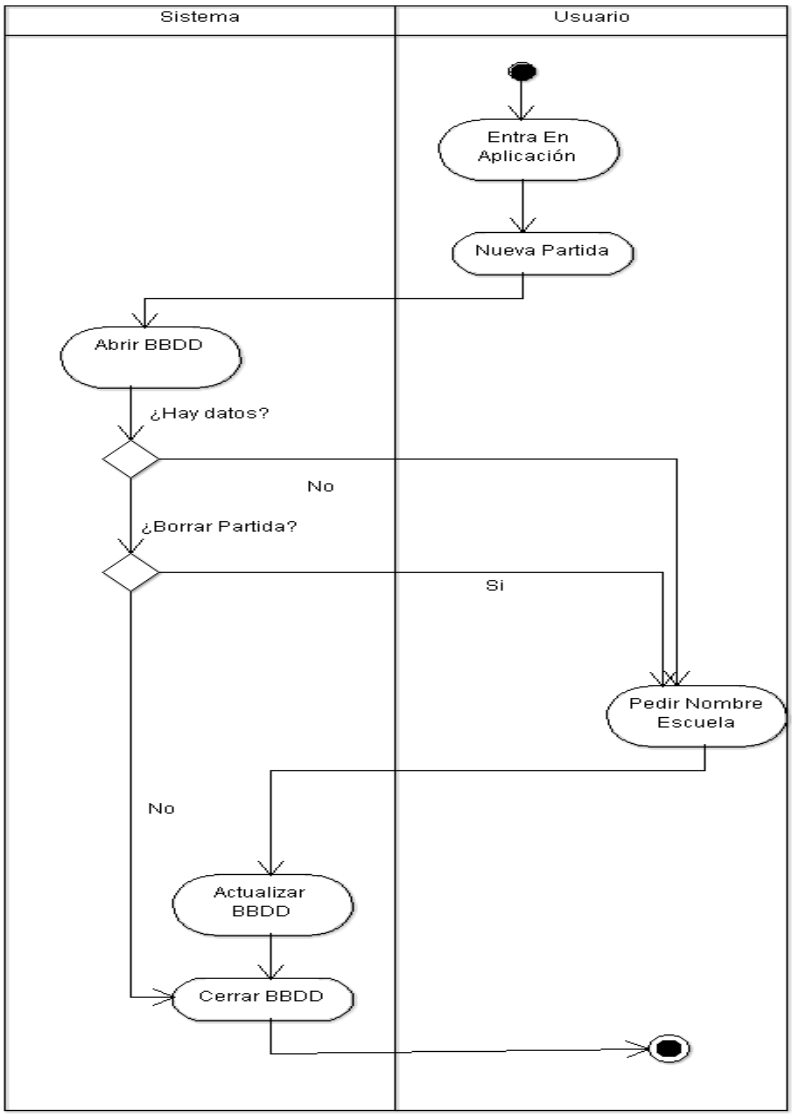


Fig. 18. Diagrama de Actividad de Nueva Partida

- *ActivityOpciones.java*

  Permite acceder al menú de opciones para controlar brillo, sonido o vibración.

- *ActivityPerdedor.java*

  Similar al caso de *ActivityGanador*, pero en este caso significa que el jugador ha perdido el combate, cambiando el aspecto de la actividad.

- *ActivityVender.java*

  Esta clase permite vender los gladiadores que ha adquirido el jugador previamente. Para que el usuario pueda vender un gladiador, tendrá que

seleccionarlo de su lista de gladiadores y darle al botón *vender*, donde entrará en ejecución el algoritmo de lógica difusa, y se mostrará mediante un mensaje el número de monedas que va a obtener por la venta. Su diagrama de clases puede verse en la Fig.19.

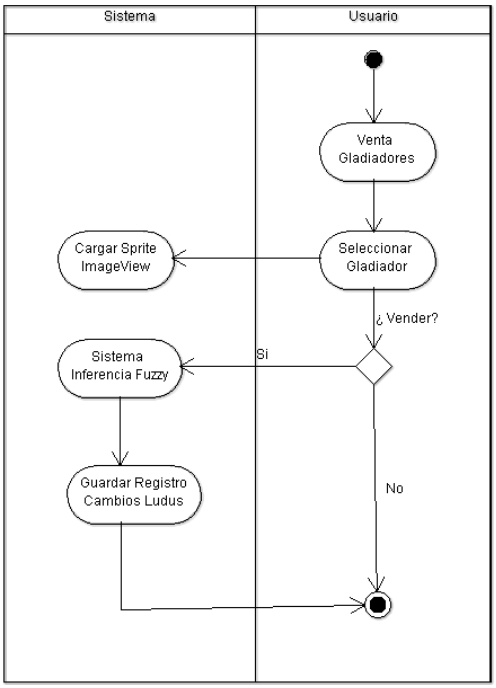


Fig. 19. Diagrama de Actividad de ActivityVender

- *ArrayEmpleados.java*

  En esta clase se crea la clase *Empleados*, que serán los objetos que insertemos en el array para el *itemAdapter* de la activity *Empleados*.

- *ArrayInstalaciones.java*

  Similar a la clase *ArrayEmpleados* salvo que esta vez creamos objetos para el Array de Instalaciones.

- *ClaseCanvas.java*

Clase principal para poder pintar todas las imágenes del juego, como pueden ser los gladiadores, armaduras, armas, etc.

- *ImageAdapter.java*

  Clase donde se crea el *ImageAdapter*, que contendrá un array de imágenes.

- *ItemAdapter.java*
  En esta clase se mostrarán al usuario los distintos empleados susceptibles de ser mejorados, siempre y cuando disponga de los recursos necesarios (dinero, nivel, etc.). Se usa un rating bar (pequeñas estrellas que cambian de color según el nivel de dicho empleado) para mostrar el nivel gráficamente.

- *ItemAdapterIns.java*

  Similar a la clase anterior, pero relativo a las instalaciones del ludus.

- *MainActivity.java*

  Pantalla inicial nada más abrir la aplicación, que muestra la opción de comenzar una nueva partida, continuarla o modificar la configuración del teléfono. Como puede verse en la Fig.20.

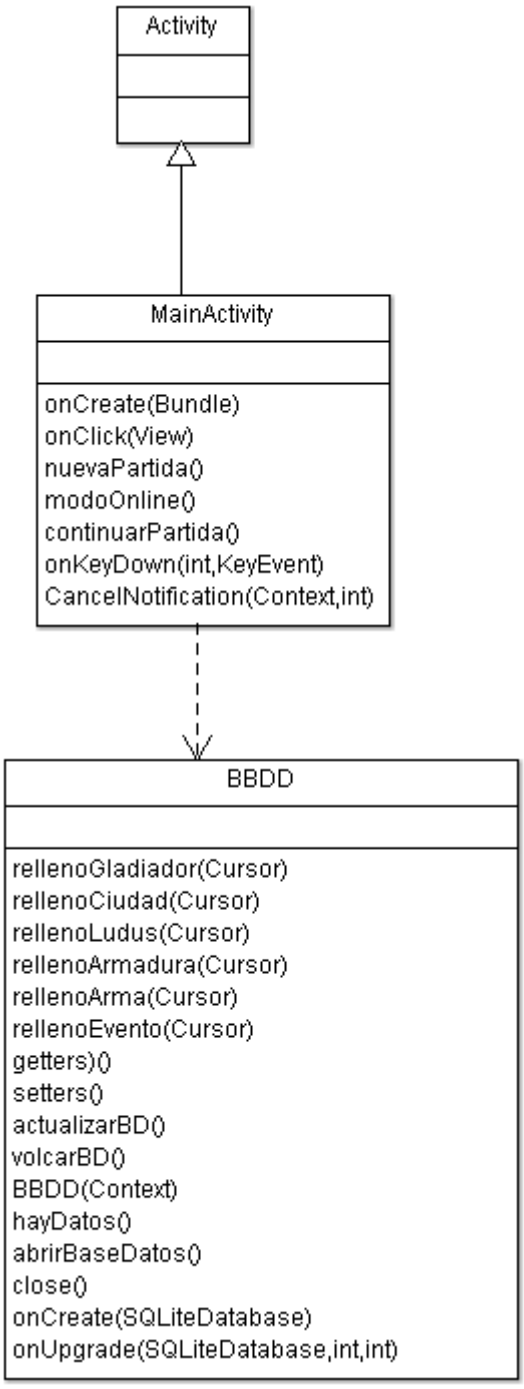


Fig. 20. Diagrama de clases de MainActivity

- *TouchImageView.java*

  Esta clase permite hacer zoom sobre el mapa. Mediante una matriz, se va trasladando la distancia deseada, consiguiendo el efecto del zoom sobre la imagen.

Algoritmos

Las distintas clases java que están contenidas en el paquete *JavaCode* son el núcleo de la aplicación, ya que se ocupan de gestionar y controlar todos los cambios y accesos que se quiera hacer en cualquier objeto de la partida (gladiadores, ludus, eventos, etc.). También contiene la clase que gestiona la base de datos, y permite volcar los datos de la memoria a la base de datos y viceversa. Se puede ver con más detalle en la Fig.21.

Fig. 21. Clases del paquete Reglas

- *Arma.java*

  Clase básica para tratar cambios o accesos a los atributos de cualquier objeto Arma. Dispone de dos constructoras, una de las cuáles se utiliza para crear en memoria las armas que se encuentren en la base de datos.

- *Armadura.java*

  Misma función que la clase anterior, pero relativa a los objetos de tipo Armadura.

- *BBDD.java*

  Esta clase extiende a su vez la clase *SQLiteOpenHelper*, la cual ofrece multitud de opciones a la hora de gestionar una base de datos de tipo *SQLite*. Contiene todos los métodos necesarios para la creación, lectura y escritura de datos en la base de datos.

- *Ciudad.java*

  Clase que gestiona el objeto más importante de la aplicación, la ciudad. Contiene referencias a todas las demás clases del paquete *JavaCode* y métodos para la inicialización, acceso y gestión de la ciudad y todos sus componentes. Se le ha aplicado el patrón *Singleton*, para asegurar que solamente existe una instancia de *Ciudad* al mismo tiempo.

- *Combate.java*

  En ésta clase se genera un combate entre dos gladiadores tomados como entrada, y devuelve el resultado del mismo. El combate se rige por un conjunto de reglas, que han sido desarrolladas de forma original, aunque basadas en la idea del popular juego de rol *D&D* [14] (se tienen en cuenta los atributos de los gladiadores, un componente de azar, y una variable "moral" que otorga mayor realismo y dinamismo a cada combate). Realiza los cambios en los atributos de los gladiadores en función de dicho resultado, y genera una lista de enteros que se usarán en el paquete *Animaciones* para reproducir fielmente mediante Sprites el combate que acaba de producirse.

- *Entrenamiento.java*

  Esta clase se ocupa de gestionar el entrenamiento de cada uno de los gladiadores, teniendo en cuenta su entrenamiento semanal asignado, sus atributos, el nivel de instalaciones de la escuela a la que pertenezcan y un componente de azar. Realiza los cambios pertinentes en los atributos de cada luchador y devuelve un resumen de las mejoras producidas en cada entrenamiento.

- *EventoCaracteristicas.java*

  Clase básica que realiza las mismas funciones que las clases *Arma.java* y *Armadura.java*, pero relativa a la gestión de los eventos.

- *Gladiador.java*

  Clase que se ocupa de generar aleatoriamente gladiadores para el mercado. También contiene métodos necesarios para cualquier cambio o acceso que se necesite realizar en los atributos de un gladiador. Por último, contiene métodos que devuelven una lista de enteros utilizada por el paquete *Animaciones* (el cuál se explica a continuación), a la hora de gestionar qué imágenes se cargarán en el *Canvas* para reproducir los combates.

- *Ludus.java*

  Esta clase contiene los métodos necesarios para la inicialización de un *ludus*. También contiene métodos imprescindibles a la hora de gestionar cualquier aspecto del mismo (entrenamientos, mejora de instalaciones o equipamiento automático de gladiadores rivales).

- *Opciones.java*

  Clase donde se puede modificar la luminosidad de la pantalla del dispositivo móvil. También se puede activar o desactivar la vibración.

Animaciones

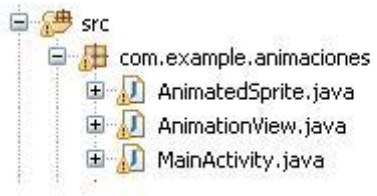


Fig. 22. Clases del paquete Animaciones

A la hora de implementar el sistema de animaciones del juego, hemos decidido utilizar las librerías internas que nos facilita Android, y no implementar librerías externas. Un ejemplo de estás librerías externas que probamos fue *Cocos2d* [15], que permite diseñar unos gráficos y animaciones más trabajados, pero debido a su complejidad y escaso desarrollo, preferimos centrarnos en explotar las librerías internas de Android para diseñar nuestras animaciones.

Mediante el uso de la técnica *Spriting*, que se explica en *el capitulo5. Técnicas Especiales Usadas*, creamos nuestras imágenes en formato *png*, las cuales, una vez introducidas en las clases previamente expuestas, nos permiten generar un gran número de modelos de combate. Las clases usadas pueden verse en la Fig.22, y su diagrama de clases en la Fig.23.

A continuación se muestra un breve resumen de la funcionalidad de cada clase.

- *MainActivity.java*

  Es la actividad principal del paquete, y desde donde se genera la interfaz gráfica de la animación. Mediante el paso de variables entre *activities*, esta clase recibe toda la información necesaria a través de los paquetes *com.example.gladiadores* y *com.javacode*.

- *AnimationView.java*

  Esta clase contiene el algoritmo para la ejecución de las animaciones. Debido a la gran carga de imágenes y el alto consumo de memoria que requiere, se ha implementado este algoritmo mediante el uso de *threads* auxiliares, liberando así al *thread* principal del programa. Una vez seleccionados los distintos *sprites* que se van a tener que cargar, información que es transmitida a través de la clase *ActivityCombate*, se pasa al algoritmo de selección del movimiento de los

gladiadores. Usando un *ArrayList*, el cual nos informa sobre qué movimiento le debe hacer a continuación cada gladiador, junto con la función *updatephysics()* del *thread*, se van cargando y mostrando los distintos movimientos en la pantalla del dispositivo.

- *AnimatedSprite.java*

  Esta es la clase donde se generan los objetos *Sprite*. Cada *png* se va recortando según un tamaño estipulado, para formar un número de *frames* de la animación. Según se van pintando de forma continuada gracias a la clase anterior, da la impresión de estar viendo una imagen en movimiento.

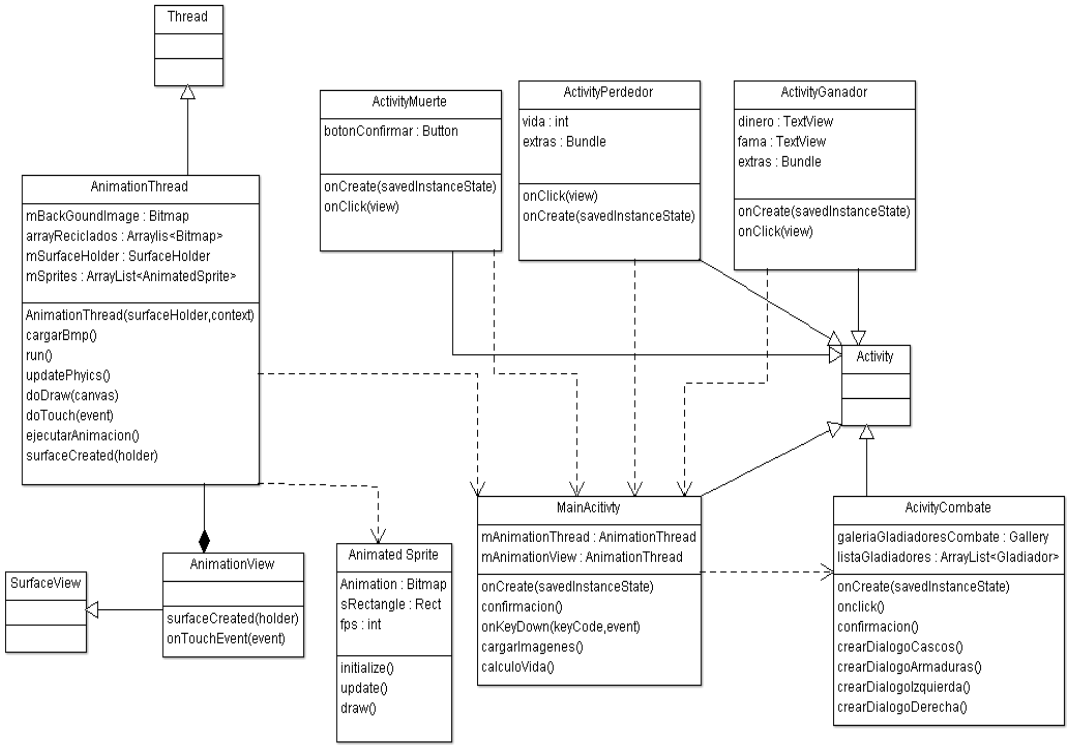


Fig. 23. Diagrama de clases Animaciones

CAPÍTULO 5. TÉCNICAS ESPECIALES UTILIZADAS

5.1 Spriting

La técnica del *Spriting* consiste en crear gráficos 2D parcialmente transparentes que pueden ser usados en videojuegos (se llaman *Sprites*) o en cualquier otro ámbito gráfico (*Pixel Art*). Se ha decidido usar éste tipo de gráficos para implementar las animaciones de combates y los modelos de personajes debido a su relativa facilidad de diseño (si se compara con otros tipos de gráficos más modernos).

El uso de sprites en videojuegos se remonta a los años 90, época en la que surgió una generación de consolas que se centraban más en el aspecto jugable y adictivo de los videojuegos y dejaban en un segundo plano el apartado gráfico. Esta técnica permitía a los desarrolladores dedicar menos tiempo a los gráficos y centrarse más en otros aspectos, y sin embargo seguir obteniendo un nivel visual realmente bueno, como podemos observar en la *Fig.24*.

Fig. 24. Sprite de Super Mario (SNES)

Existen varias maneras de diseñar *sprites* actualmente, en este caso hemos usado la metodología "*Line Art"*, la cual consiste en comenzar el diseño trazando su contorno mediante el uso de líneas en colores oscuros. Una vez se tenga dibujado el contorno de la figura se colorea sin ningún tipo de detalle cada una de las partes que definen la figura (piel, ropa, etc…), para después continuar añadiendo detalles más exquisitos (sombreado, arrugas de la piel o la ropa, brillos metálicos, etc.). En la *Fig.25* se pueden comprobar todos los pasos que se deben tomar en el uso de ésta metodología de diseño de *sprites*:

Fig. 25. Diseño de un sprite mediante la técnica "*Line Art"*

Para el diseño de sprites existen varios programas que hacen mucho más sencilla y fluida la tarea. Algunos como *GraphicsGale* o *Pro Motion* están específicamente diseñados para la creación de sprites y animaciones. Otros como *GIMP* y el popular *Adobe Photoshop* son de uso más general, pero ofrecen funcionalidades suficientes como para diseñar sprites sin ningún tipo de problema. Estas dos últimas herramientas han sido las utilizadas en el proyecto, sobre todo *Adobe Photoshop*, ya que la posibilidad del dibujo por capas hizo más sencillo el diseño de cada nuevo *sprite*.

Para este proyecto se han utilizado imágenes en formato *PNG*, con dos filas de cinco sprites cada una, que contienen todos los posibles movimientos de cada personaje del videojuego. En la *Fig.26* se puede observar cómo queda una de esas imágenes mientras está siendo editada por capas con *Adobe Photoshop*.

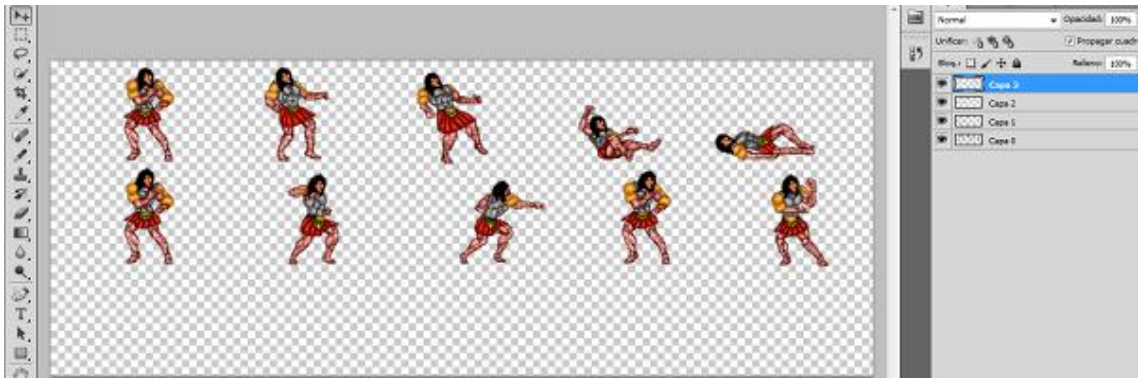


Fig. 26. Diseño de sprites por capas con *Adobe Photoshop*

Gracias al uso de las capas, es posible simular un elevado número de combates distintos, superponiendo las distintas partes que componen la fisionomía y armamento de cada gladiador. De esta forma, se genera un conjunto de imágenes diferentes para la cabeza, cascos, cuerpo, armaduras y armamento, que posteriormente se cargan indistintamente para generar un combate, según las características de ambos gladiadores. Posteriormente, debido a la funcionalidad de la clase *Canvas* que proporciona Android, la cual permite trabajar por capas, se inserta cada imagen con unas coordenadas determinadas en el *Canvas*. Al fijarse cada una en su posición, se crea la sensación de estar viendo una sola imagen para cada gladiador.

Este método, permite generar un número de combates prácticamente infinito, permitiendo, además, añadir nuevas armas, armaduras, cabezas, cuerpos o cascos, de forma rápida y sencilla. A continuación en la *Fig.27* se muestra cómo se crean las distintas capas por medio de *Adobe Photoshop* y el resultado final obtenido, que será el que visualizará el usuario durante su partida.

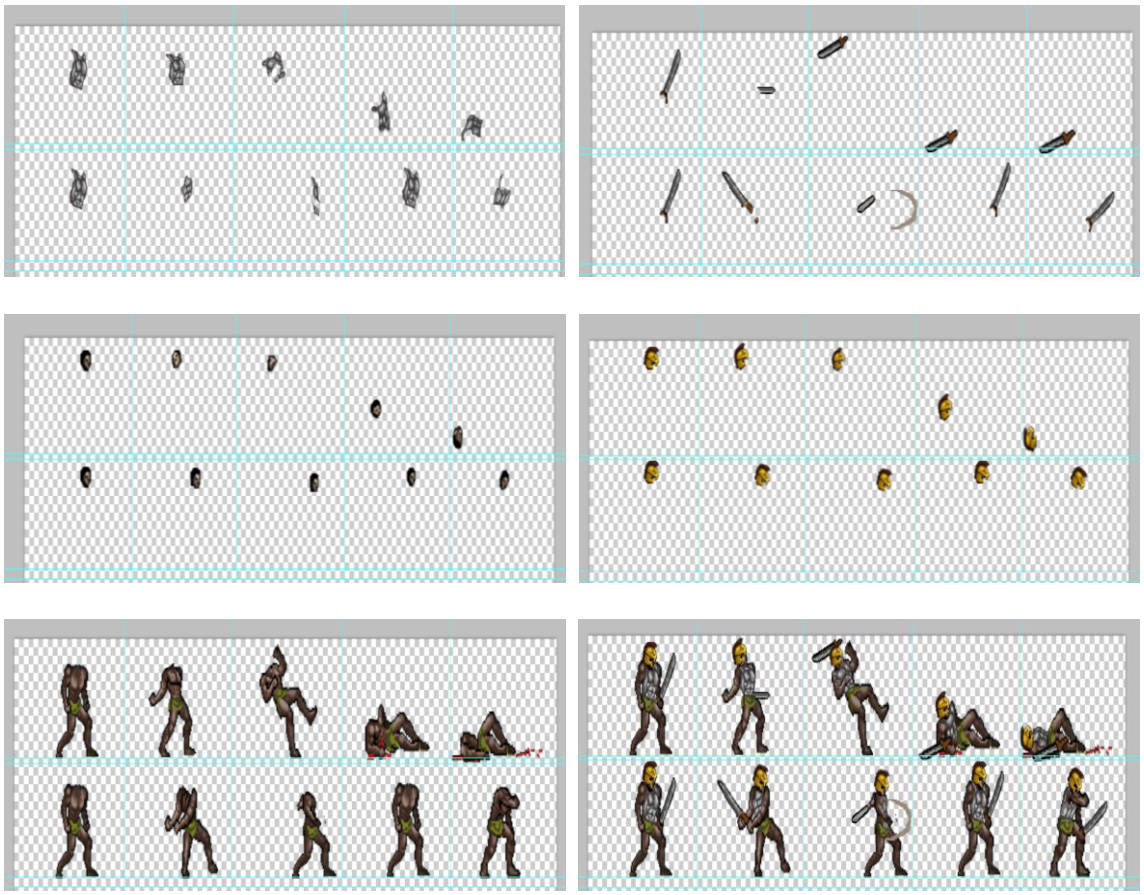


Fig. 27. Proceso de creación de gladiadores mediante Adobe Photoshop

5.2 Soft Computing

Las técnicas de soft computing se emplean para solucionar problemas cuya información conocida es incompleta, inexacta o posee cierto nivel de incertidumbre. En este trabajo se quiere dotar al sistema de juego de cierta flexibilidad, de tal forma que el usuario no se encuentre siempre con unos resultados predeterminados y predecibles. A continuación se pueden ver las distintas técnicas que existen de soft computing, se analizan sus puntos fuertes y débiles, y se comentan las razones por las cuales se ha optado por implementar un sistema de *Lógica Difusa* en este proyecto, donde se explica con más detalle, en el artículo *Ludus Manager con Lógica Fuzzy* [23].

Algoritmos Evolutivos

Se trata de una técnica basada en los mecanismos de la evolución biológica y la teoría de la evolución formulada por Charles Darwin. Estos algoritmos son métodos de optimización y búsqueda de soluciones que se utilizan en problemas con extensos espacios de búsqueda, ya que con otros métodos no se consiguen encontrar soluciones en un tiempo razonable.

Los algoritmos evolutivos más usados actualmente son los algoritmos genéticos, con mucho éxito especialmente en la Teoría de Juegos. Son métodos de búsqueda que se basan en la probabilidad, hacen evolucionar una población de individuos sometiéndola a una serie de acciones y cambios aleatorios, decidiendo cuáles de dichos individuos son los más adaptados tras los cambios, y cuáles son descartados.

El funcionamiento general de un algoritmo genético consta de los siguientes pasos:

- Inicialización: Se genera aleatoriamente una población inicial constituida por un conjunto significativo de las posibles soluciones al problema.

- Evaluación: Se aplica una función a cada solución para averiguar su calidad.

- Condición de término: El algoritmo se detiene cuando alcance una solución óptima al problema, como ésta normalmente no se conoce, se utilizan otros criterios de término que pueden ser: ejecutar el algoritmo un número determinado de veces, o detenerlo cuando no haya más cambios en la población de individuos. Mientras no se cumpla la condición de término se siguen los siguientes pasos:

    - Selección: Elegimos qué cromosomas cruzamos en la siguiente generación, los que tengan mejor aptitud tienen más posibilidades de ser seleccionados.

    - Cruzamiento: Se emparejan los cromosomas de dos en dos, generando otros dos descendientes combinando las características de sus padres.

    - Mutación: Modificamos aleatoriamente parte del cromosoma de cada individuo.

    - Reemplazo: Seleccionamos los mejores individuos de la población, que pasan a la siguiente generación.

Redes Neuronales

Se trata de una técnica basada en la forma en que funciona el sistema nervioso de los animales. Un sistema de neuronas interconectadas entre sí, que colaboran para producir una salida. Cada una de las neuronas recibe varias entradas y emite una salida, que viene dada por tres funciones:

- De propagación o excitación: Se suma cada entrada recibida y se multiplica por el peso que tiene su interconexión, si éste es positivo se habla de una conexión excitadora, sino, se denomina inhibitoria.

- De activación: Modifica la función anterior.

- De transferencia: Se aplica al valor que devuelve la de activación. Acota la salida según la interpretación que le demos a dichas salidas.

Esta técnica posee varias ventajas, como por ejemplo la habilidad de aprender durante una etapa de aprendizaje, en la cual proporcionamos a la red unos datos de entrada y se le indica cuál es la salida que esperamos. Sin embargo, también existen tipos de *Redes Neuronales* que se rigen por modelos de aprendizaje no supervisado, que no precisan de ningún conjunto de entradas previo.

Lógica Difusa

Esta ha sido la técnica elegida para implementar en nuestro proyecto, debido a la gran utilidad que nos aporta, ya que ofrece soluciones para sistemas de decisión en general, resolución y comprensión de datos, y es capaz de imitar la forma en la que tomamos decisiones los humanos. El ámbito donde se aplica dicha técnica es en el cálculo del precio de venta de los gladiadores que adquiera el usuario. A continuación se explica mejor su funcionamiento con el ejemplo siguiente:

En la lógica binaria convencional, se puede desarrollar un sistema que analize características de un jugador de baloncesto, y devuelva la calidad de ese jugador mediante un valor numérico comprendido entre 0 y 10. Se puede tener el conjunto de reglas siguiente:

Si un jugador mide más de 210cm, se devuelve 2, si su estatura está comprendida entre 195cm y 210cm, se devuelve 1, y se devuelve 0 en otro caso.

Si la edad de un jugador está comprendida entre 22 y 28 años devuelve 2, si está comprendida en los intervalos 18-21 y 29-32 devuelve 1, y se devuelve 0 en otro caso.

Si el porcentaje de acierto en tiros de 3 puntos es superior al 50% se devuelve 2, si es inferior al 20% se devuelve 0, y devuelve 1 en cualquier otro caso. Creamos otras dos reglas exactamente iguales que la anterior, pero con los tiros de 2 puntos, y los tiros libres de 1 punto.

Una vez hecho esto, se toman como entrada dos jugadores con los siguientes datos y porcentajes:

Jugador 1: 194cm, 21 años, 48% tiros de 3,80% tiros de 2, 100% tiros libres de 1 punto.
Jugador 2: 195cm, 28 años, 5% tiros de 3,50% tiros de 2,60% tiros libres de 1 punto.

Tras esto se aplican las reglas propuestas anteriormente, y dan como resultados numéricos respectivamente 6 y 7. A simple vista, se observa que el jugador 1 es mejor que el jugador 2. Entonces, ¿por qué el sistema indica que es mejor jugador el segundo?

El problema radica en que el sistema no es capaz de entender que la diferencia entre alturas es prácticamente irrelevante, que las edades de ambos jugadores están próximas al siguiente intervalo, el porcentaje de tiros de 3 del segundo jugador es bastante malo, o los porcentajes del primer jugador son bastante buenos. Para solucionar

estas incongruencias, la lógica difusa tiene en consideración y comprende los cuantificadores que usamos los humanos en nuestro lenguaje. Por tanto, el primer jugador tendría un porcentaje de tiros libres absolutamente perfecto, y el segundo jugador tendría un porcentaje de tiros libres del que se diría que es aceptable.

La justificación y sobre todo las ventajas de usar este tipo de técnica en un proyecto informático son, entre otras, su rápida implementación, lo flexible a cambios y lo aplicable que es a cualquier otra tecnología. También se puede ver el estudio que ya se ha realizado sobre la utilización de la lógica difusa en este tipo de dispositivos en [15-17]

5.3 Implementación del Sistema de Inferencia Difuso (FIS)

El esquema desarrollado para el sistema de inferencia fuzzy que se utiliza para calcular el precio de venta de un gladiador se puede ver en el diagrama de actividad de la Fig.28, y consta de los siguientes elementos y procesos:

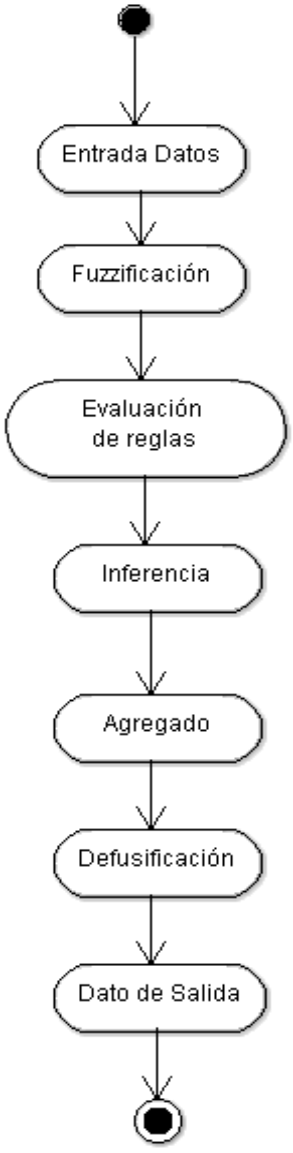


Fig. 28. Diagrama de Actividad del FIS.

1. **Entrada Datos**: Se trata de tres datos numéricos que provienen de la clase *Gladiador*, cuyos valores representan la media de las características físicas, habilidades que tenga en la lucha y la fama de un personaje del juego.

2. **Fuzzificación**: Esta es la fase de conversión de los datos de entrada en valores correspondientes a las funciones de pertenencia de acuerdo con el modelo difuso. La función de pertenencia, que nos indica el grado de pertenencia de una variable al conjunto correspondiente, viene dada por el fragmento de código en la Fig.29. En la Fig.30, se muestra la definición de los conjuntos.

```
public double  fuzzificacion(double valor)
{
    if(valor<valorEjeX[0] || valor>valorEjeX[3])
        return 0;

    if(valor>=valorEjeX[1] && valor<=valorEjeX[2])
        return 1;

    if(valor>=valorEjeX[0] && valor<valorEjeX[1])
        return  (valor-valorEjeX[0])/(valorEjeX[1]-valorEjeX[0]);

    if(valor>valorEjeX[2] && valor<= valorEjeX[3])
        return (valorEjeX[3]-valor)/(valorEjeX[3]-valorEjeX[2]);

    return 0;
}
```

Fig. 29. Método Trapezoidal.

```
VariableLinguistica caracteristicas = new VariableLinguistica("caracteristicas");
caracteristicas.add("bajo",0,0,60,75);
caracteristicas.add("alto",65,80,150,150);
```

Fig. 30. Código de definición de la variable lingüística "Características".

3. **Evaluación de reglas**: Se define un total de ocho reglas, resultado de la evaluación de dos etiquetas (bajo y alto) para tres variables de entrada (características, habilidades y fama). El conjunto de reglas se muestran en la Fig. 31.

```
if caracteristicas is bajo and habilidades is bajo and fama is bajo then precio is bajo
if caracteristicas is bajo and habilidades is bajo and fama is alto then precio is medio
if caracteristicas is bajo and habilidades is alto and fama is bajo then precio is bajo
if caracteristicas is bajo and habilidades is alto and fama is alto then precio is medio
if caracteristicas is alto and habilidades is bajo and fama is bajo then precio is medio
if caracteristicas is alto and habilidades is bajo and fama is alto then precio is alto
if caracteristicas is alto and habilidades is alto and fama is bajo then precio is medio
if caracteristicas is alto and habilidades is alto and fama is alto then precio is alto
```

Fig. 31. Código de implementación del conjunto de reglas.

El operador lógico que se ha utilizado para la conjunción entre los conjuntos es el mínimo.

$$\mu A \cap B(x) = \min[\mu A(x), \mu B(x)]$$

4. **Inferencia**: La inferencia aplicada a las reglas determina el conjunto de salida de cada regla por medio del operador de Mamdani por mínimos.

$$\min(\mu, \mu W(z)), \forall z$$

5. **Agregado**: Finalmente, obtenemos la función de pertenencia de la variable de salida, que en nuestro caso es a partir de la operación MAX entre todos los conjuntos de salida de la etapa de inferencia. Con esto se consigue que el precio de venta del gladiador sea el máximo posible. La implementación de esta agregación en Java puede verse en la Fig. 32.

```
if (currentToken.equalsIgnoreCase("bajo") && tempResult>minimo ){
    v.setMinimoConjuntosBajo(tempResult);
}
else if (currentToken.equalsIgnoreCase("medio") && tempResult>medio  ){
    v.setMinimoConjuntosMedio(tempResult);
}
else if (currentToken.equalsIgnoreCase("alto") && tempResult>maximo ){
    v.setMinimoConjuntosAlto(tempResult);
}
```

Fig. 32. Implementación de la agregación en Java.

6. **Defusificación**: Se determina el dato más representativo mediante la técnica del centroide basado en el cálculo discretizado de la fórmula en rango real. Su código se muestra en la Fig. 33.

```
//Metodo del centroide
double nominador = 0.0;
double denominador = 0.0;
for(int i=0; i<100; i++)
{
     nominador+=(minimo+pasos*i)*sum[i];
     denominador+=sum[i];
}

return nominador/denominador;
```

Fig. 33. Implementación Java del método del Centroide.

Un ejemplo de evaluación de 3 reglas, fuzzificación, reglas, inferencia y agregado se puede ver en la Fig.34.

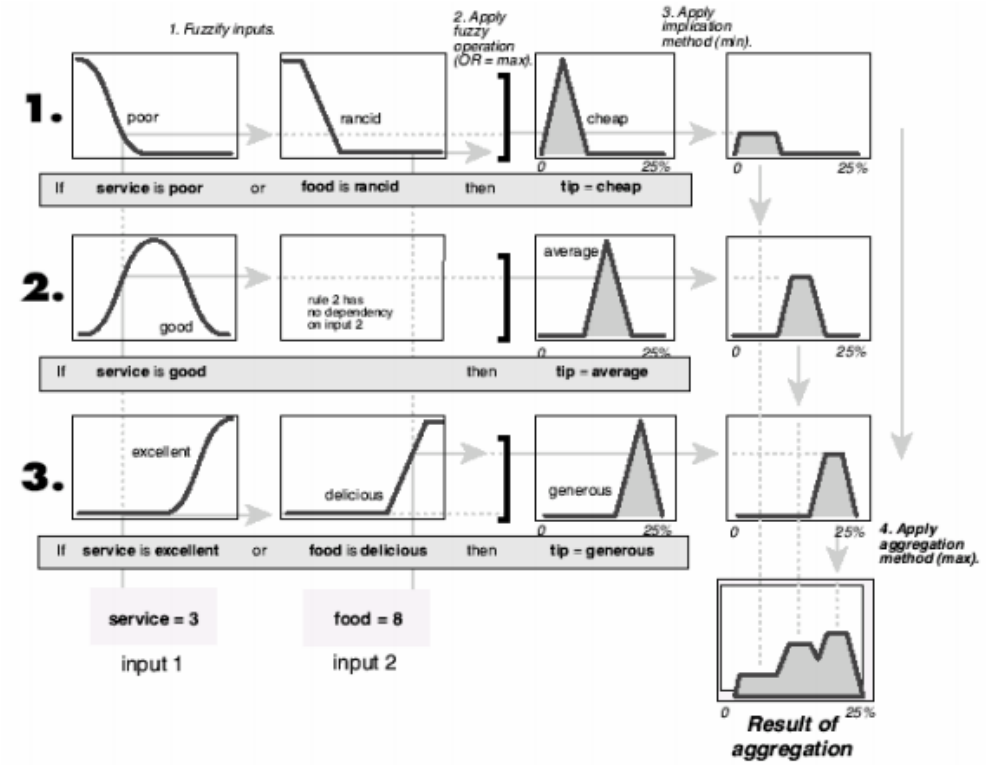


Fig. 34. FIS con 3 reglas.

CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO

6.1-Trabajo Futuro

La principal evolución que se plantea de la aplicación es incluir un modo online multi-jugador, donde múltiples usuarios puedan enfrentarse entre sí. Un posible modelo de arquitectura sería la que se muestra en la Fig.35. En él, un servidor de bases de datos almacenará la información relevante sobre los usuarios y las partidas multi-jugador. El *ServidorAplicacionesAndroid* se ocupará de gestionar las peticiones de los usuarios durante las partidas con varios participantes, así como de realizar las peticiones de consulta y actualización necesarias al *ServidorBBDD*.

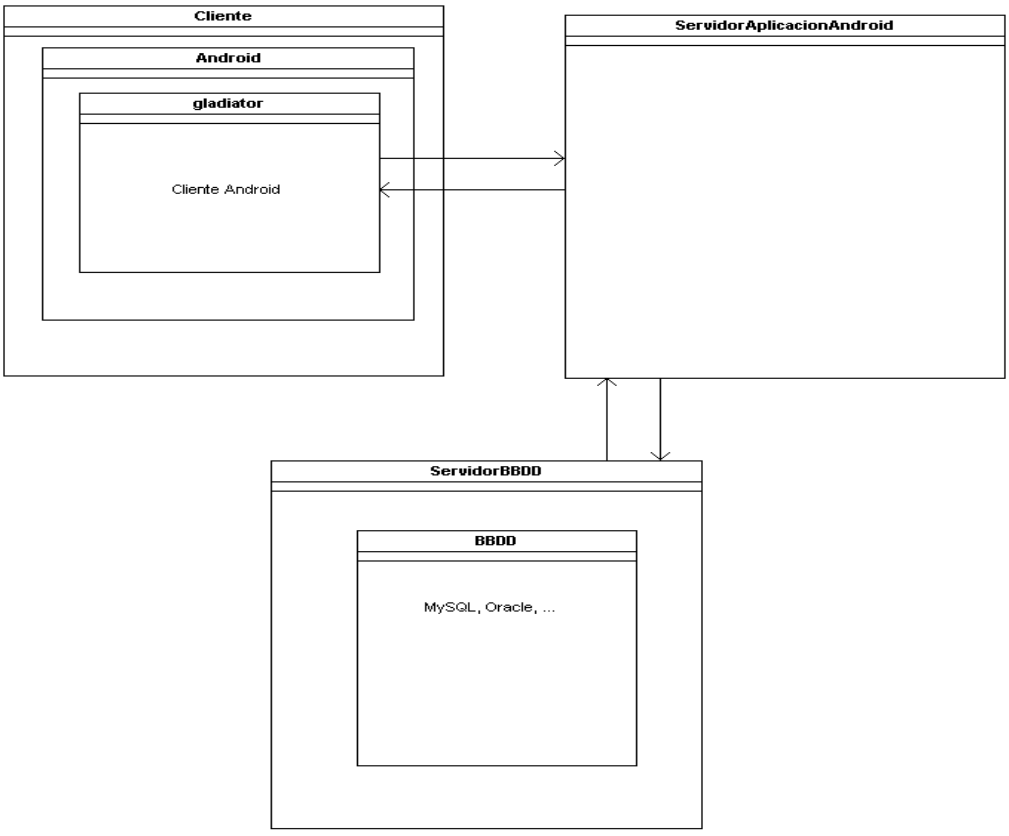


Fig. 35. Arquitectura modo online.

En relación con el juego online, se ofrecerá al jugador la posibilidad de registrarse antes de comenzar a jugar. Con esto se consigue que el usuario esté identificado cuando se conecte, y mantener la privacidad de la partida y datos del jugador.

La arquitectura planteada para el juego online permitirá además abordar otros requisitos que ampliarán la funcionalidad del juego, consiguiendo que sea más completo y atractivo. Entre estos requisitos adicionales se encontrarían un ranking de

puntuaciones de los usuarios online y sus combates con otros jugadores, así como la posibilidad de pedir consejo a otros jugadores en determinadas circunstancias del juego como combates, mudanzas y compras de gladiadores. Como aliciente para el juego online, se podrían incluir apuestas de gladiadores, donde los jugadores enfrentarán a dos gladiadores de sus escuela entre sí en un combate. El vencedor, se vería recompensado con la adquisición del luchador rival.

Otro posible requisito sería la ya mencionada opción de mudarse a otra ciudad. Como se comentó anteriormente, esto implicaría que el jugador vendiera su escuela y comprara otra en una ciudad de mayor importancia. El jugador podría llevarse consigo sus gladiadores, empleados y equipamiento a la nueva escuela. Este requisito, implica a su vez la opción de incorporar un mapa con las distintas ciudades y las escuelas que tengan. Siguiendo el hilo de esta propuesta, la inclusión de varias escuelas rivales sería otra característica a tener en cuenta. Cada escuela estaría ubicada en una ciudad diferente, y no solo combatirían contra nosotros, sino también entre ellas. La inteligencia artificial de cada rival sería diferente, posibilitando una curva de aprendizaje del juego mucho más ajustada, ya que los enfrentamientos contra las escuelas con Inteligencia Artificial más avanzada se dejarían para más adelante.

La aplicación desarrollada en este proyecto tiene un intencionado aspecto gráfico 2D similar al de los antiguos videojuegos. Existe una tendencia de cierto sector de usuarios a recordar los juegos a los que jugaban en su infancia. Sin embargo, hay otros, que prefieren una experiencia gráfica más próxima a los modernos videojuegos y que explote las características de los terminales actuales. Para evolucionar los gráficos del juego según esta demanda, se podrían implementar por medio de librerías gráficas con jPCT [19] u OpenGL ES[20], o bien utilizando frameworks actualmente en desarrollo para OpenGL como pueden ser Dwarf-fw [21] o Forget3D [22]. De todas formas, si el diseño en 2D tuviera éxito, siempre cabría la opción de mejorar las animaciones, insertando más *frames*, diseños más trabajados, y una mejoría gráfica en general.

Para aumentar las características y dinamismo de los combates, se podría incluir una opción de estrategia de combate, donde el jugador podrá elegir el estilo de lucha de su gladiador: defensivo, conservador, agresivo, etc. Esto ayudaría a reforzar la idea de un juego de estrategia, donde cada pequeña decisión que tome el usuario puede ser de vital importancia.

Incluir nuevas armas y equipación también sería importante para alargar la vida del juego, así como nuevos diseños de gladiadores y nuevas características.

6.2-Conclusiones

Una vez desarrollado este proyecto de fin de carrera, podríamos obtener diversas conclusiones. Nuestro objetivo era desarrollar un videojuego competitivo, donde la toma de decisiones por parte del usuario fuese decisiva. Esta funcionalidad ha sido abordada con éxito, ya que, si el usuario juega de manera precipitada y sin utilizar una cierta coherencia o lógica en sus acciones, esto supondrá el fracaso de la gestión de su escuela.

En cuanto a la lógica difusa, tras finalizar el proyecto se puede subrayar la gran importancia que tienen los mecanismos para gestionar el conocimiento de las personas para llevar a cabo la dirección de proyectos o negocios. La lógica difusa es un mecanismo para reflejar estas reglas de toma de decisiones.

Durante el desarrollo de este proyecto hemos obtenido y mejorado nuestros conocimientos sobre distintos lenguajes de programación o software. En los primeros meses no teníamos apenas experiencia con el sistema operativo Android, ya que hasta entonces nunca habíamos trabajo con él. A día de hoy somos capaces de desarrollar con facilidad un proyecto de cierta complejidad en dicho sistema operativo, haciendo buen uso de todos sus componentes, desde XML, hasta el uso de animaciones, hilos o conexiones con bases de datos. Al haber introducido lógica difusa, para comprobar que los resultados obtenidos eran aceptables, hemos utilizado el programa de programación matemática *MatLab*, en el cual somos capaces de implementar un sistema de lógica difusa completo. Para desarrollar la técnica de *spriting* hemos obtenido un vasto conocimiento en programas de tratamiento de imágenes como puede ser *Adobe Photoshop* o *GIMP*.

Al ser un proyecto formado por varios integrantes, hemos visto potenciada nuestra habilidad para dividir y organizar un trabajo de cierta envergadura. Se ha asignado a cada uno de los componentes una carga de trabajo similar y que potenciase sus conocimientos en las distintas áreas del desarrollo.

REFERENCIAS

1. Pro|Chile, Estudio de mercado de videojuegos para dispositivos móviles. (2011) http://es.scribd.com/doc/132409123/Estudio-Mercado-Juegos-Moviles
2. IDC. http://www.idcspain.com/about/index.jsp
3. Android Market, https://play.google.com/store
4. POU, http://www.pou.com.es/tag/pou-juego-gratis/
5. Photoshop-http://www.softonic.com/s/photoshop-cs6-espa%C3%B1ol
6. Eclipse http://www.eclipse.org/
7. MatLab http://www.mathworks.es/
8. Open Source: http://es.wikipedia.org/wiki/C%C3%B3digo_abierto
9. GIMP http://www.gimp.org.es/
10. Android, http://www.android.com/
11. Componentes Android, http://developer.android.com/guide/components/index.html
12. Interfaces, http://developer.android.com/guide/topics/ui/index.html
13. Sistema Inferencia Mamdani, http://www.dma.fi.upm.es/research/FundMatSoftComputing/fuzzyinf/introfis.htm
14. D&D Manual del Jugador y Guía del DM (Versión 3.5). Escrito por Wizards of the Coast, 2003.
15. Cocos2D, http://www.coco2d.org
16. Maria Alice P. Jacques, MattiPursula, JarkkoNiittymäki, and IisakkiKosonen."The impact of different approximate reasoning methods on fuzzy signal controllers"
17. Mamdani, E. H., Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis, IEEE Transactions on Computers, Vol.C-26 (12) pp: 1182 – 1191, (1977).
18. Sazonov, E. S. et al. (2002) Fuzzy logic expert system for automated damage detection from changes in strain energy mode shapes, Non-Destructive Testing and Evaluation, Taylor & Francis Publishing, Volume 18, Number 1, Pages 1 – 17
19. jPCT. http://www.jpct.net/
20. OpenGL ES. http://www.khronos.org/opengles/
21. Dwarf-fw. https://code.google.com/p/dwarf-fw/
22. Forget3D. https://code.google.com/p/forget3d/
23. Domínguez, A, Iñigo, J., Morata, M., López, V. y Fuentes, R. Modelado de Ludus Manager con Lógica Fuzzy, Actas del Congreso CEDI 2013, pendiente de publicación.

APÉNDICE- MANUAL DE USUARIO

Al abrir la aplicación de Ludus Manager, el usuario se encontrará con la pantalla principal, que se muestra en la Fig.1. El jugador tendrá distintas opciones, como comenzar una nueva partida, borrando la partida existente si ya había comenzado una anteriormente, continuar la partida ya empezada o configurar la luminosidad o la vibración de la aplicación. La versión online no está disponible en esta versión del videojuego, pero en una versión futura sí estará disponible. En la Figura 2 se muestran las interfaces de las distintas posibilidades que puede elegir el usuario.

Fig.1. Interfaz del menú principal

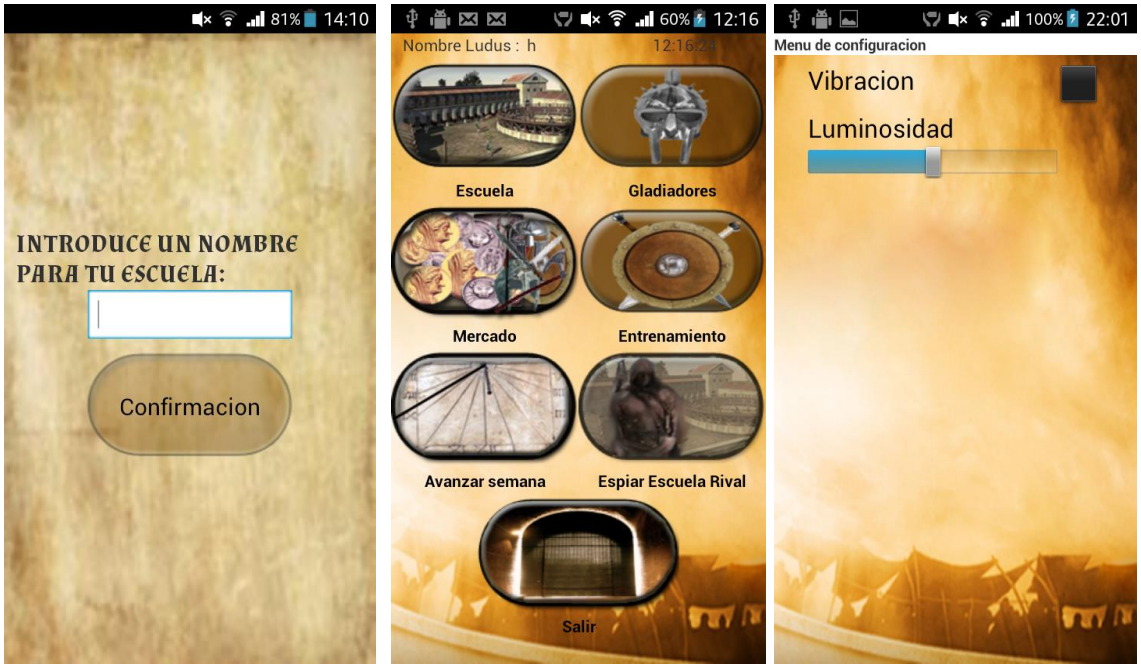


Fig.2. Nueva Partida, Opciones y Menú Principal

En el caso de comenzar una nueva partida, se pedirá un nombre para la escuela del jugador y se pasará a la interfaz de menú principal. Una vez en el menú principal, el usuario tiene las siguientes opciones:

-Consultar Escuela
- Consulta Empleados
- Consulta Instalaciones

-Consulta Gladiadores
- Características

-Mercado
- Comprar Gladiadores
- Comprar Armas
- Vender Gladiadores

-Entrenamiento
- Características

-Avanzar Semana
-Espiar Escuela Rival
-Notificaciones
-Salir

En el caso de consultar la escuela, el usuario se encontrará con una interfaz, donde tendrá toda la información de su escuela: dinero disponible, nombre de la escuela, su nivel, reputación, etc. Habrá dos botones para poder acceder a la información de los empleados e instalaciones, con la posibilidad de mejorar todas aquellas características que el usuario vea necesarias. En todo momento, pulsando el botón “menú” del dispositivo se podrá acceder a la información de la escuela.

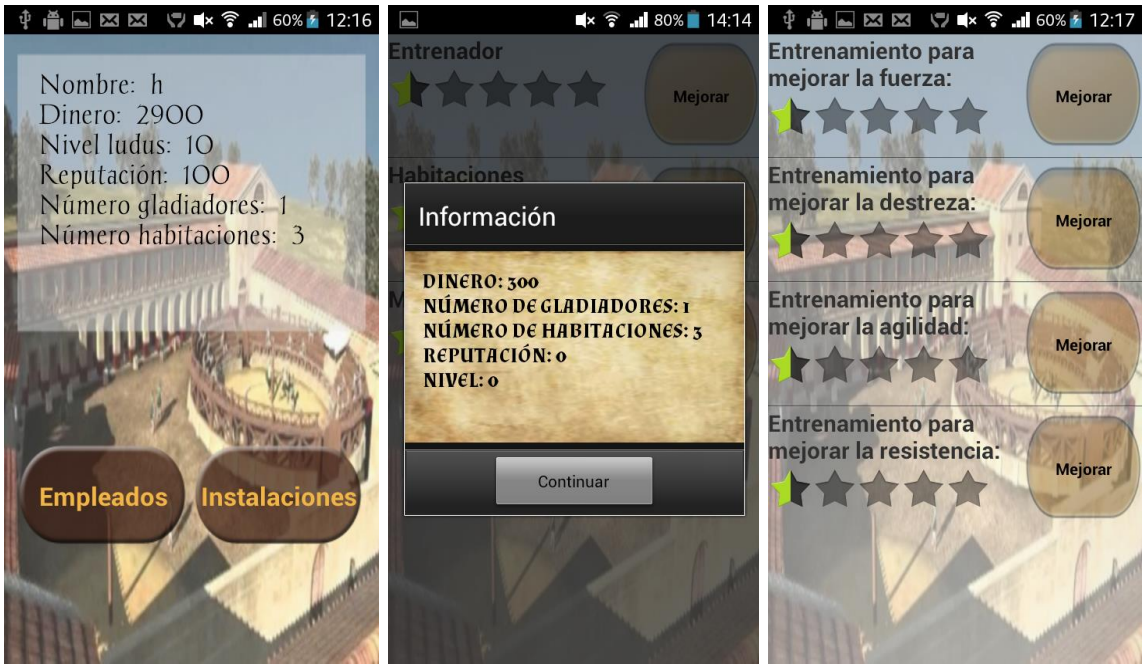


Fig.3. Consulta Escuela, Consulta Empleados, Consulta Instalaciones

Pulsando el botón de mercado se podrá acceder a la compra de armas, gladiadores, y a la venta de estos últimos. Cuando se pulsa el botón de mercado, aparecerá una caja de selección, donde se podrá acceder a los distintos tipos de mercado. Estas interfaces se muestran en la *Fig.4* y *Fig.5*. En cada una de ellas, se puede ver las características de cada gladiador o arma, con la lista que aparece en la parte de la derecha de la pantalla.

Fig.4. Selección en Mercado.

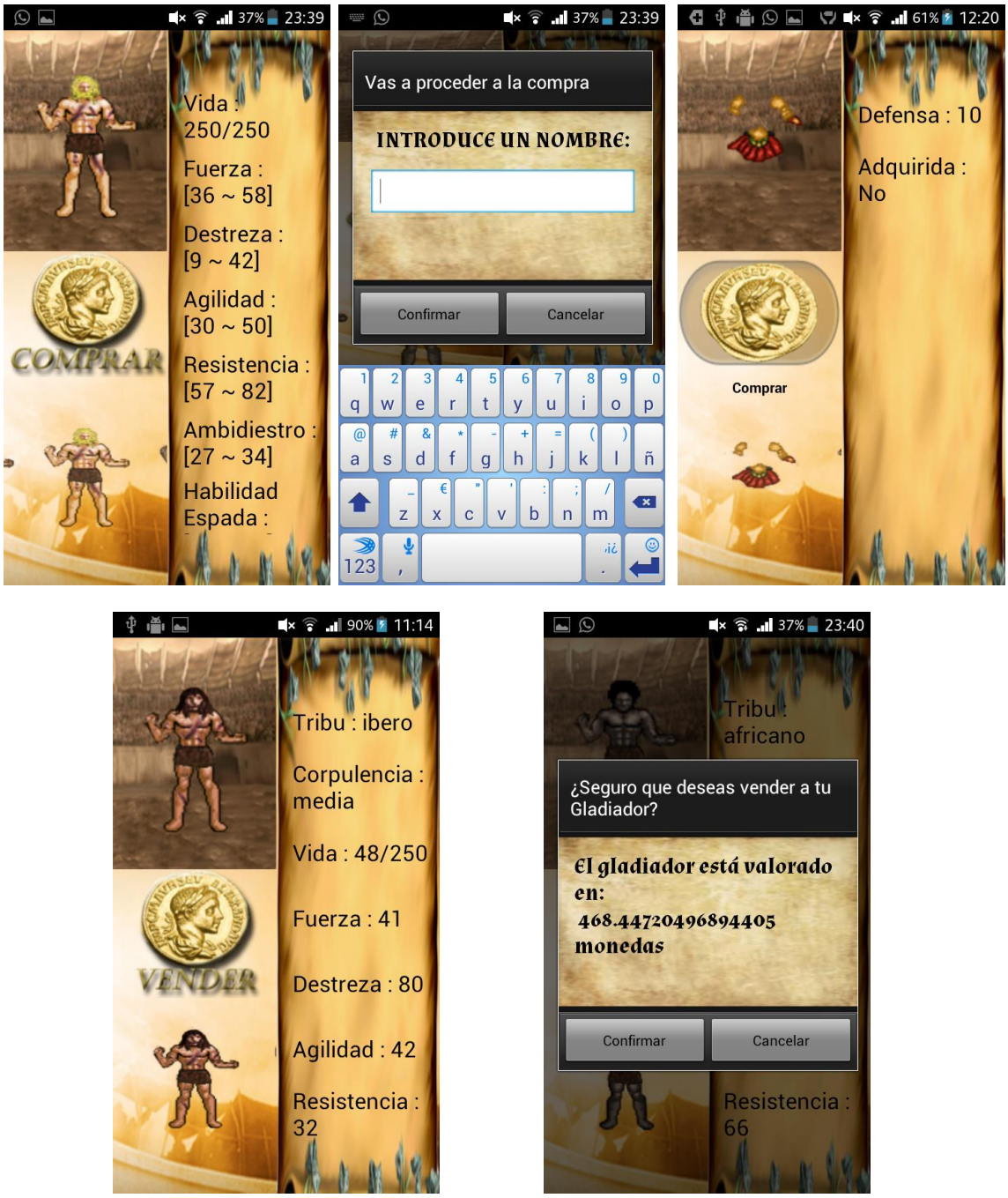


Fig.5. Mercado de Gladiadores, Mercado de Armas y Venta de Gladiadores

Una vez comprados los gladiadores, el usuario podrá consultar o entrenar aquellos gladiadores que haya adquirido. En el caso de consultar, podrá ver sus características o equipar con el armamento adquirido previamente. A través de los cuatro botones de selección, el usuario podrá elegir el armamento que más le satisfaga. Estos son, la selección de cascos, armaduras para el cuerpo, y equipamiento para cada una de las manos.

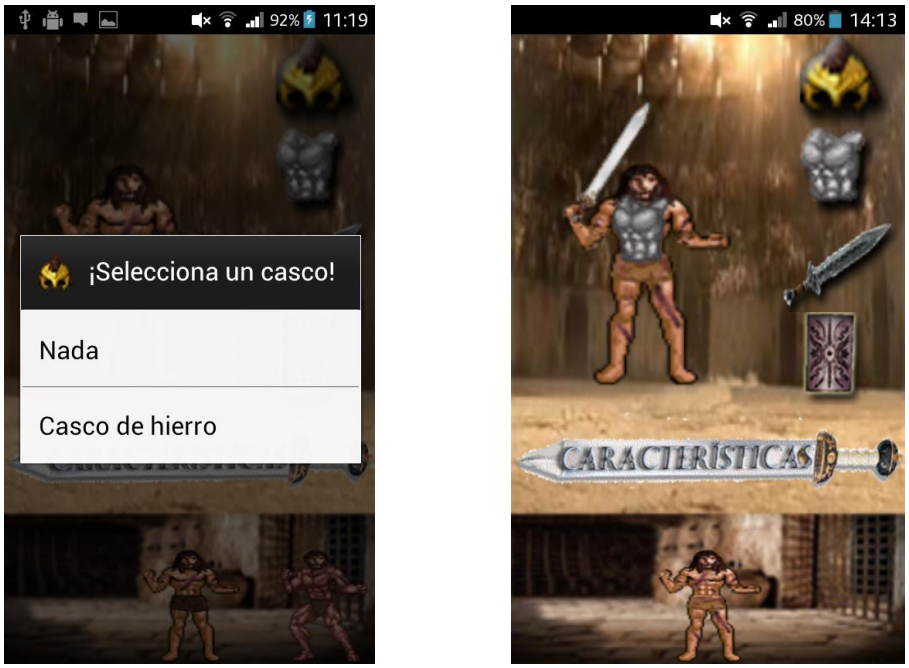


Fig.6. Consulta de Gladiadores y opciones de equipamiento

En el caso del entrenamiento, el usuario tendrá la opción de ver las características, para mejorar todo aquello que considere necesario. Dispondrá de dos botones, uno donde se mejorarán las características físicas, y otra opción, donde se podrán mejorar las habilidades de combate.

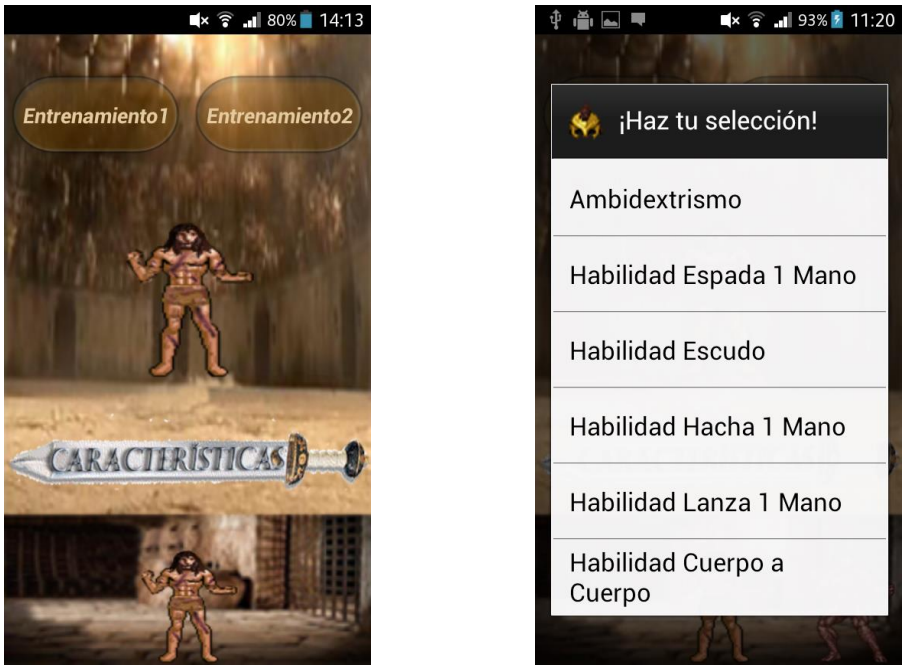


Fig.6. Interfaz de Entrenamiento

Si pulsamos la tecla *menú* de nuestro dispositivo se podrá acceder a una nueva ventana, en la que se muestran los entrenamientos seleccionados para cada gladiador.

Fig.7. Consulta de Características y entrenamientos seleccionados

Mediante el botón *Avanzar semana* avanzamos en el tiempo, lo que se traduce en diferentes tipos de cambios. Si estábamos entrenando a nuestros gladiadores, estos mejorarán, si han sido heridos, recuperarán parte de su vida, y nuestras arcas incrementarán ligeramente. Mientras, la escuela rival también aprovechará para mejorar sus instalaciones, entrenar y curar gladiadores, conseguir nuevos empleados, y demás opciones de gestión que se permite en el juego. Cada cierto número de semanas se generarán eventos, que consistirán en distintos tipos de combate (a muerte, exhibición, varios rounds, torneos, etc.) contra el ludus rival.

En caso de que se genere un evento, recibiremos un aviso en la barra de notificaciones de nuestro dispositivo, como se puede ver en la Fig.8.

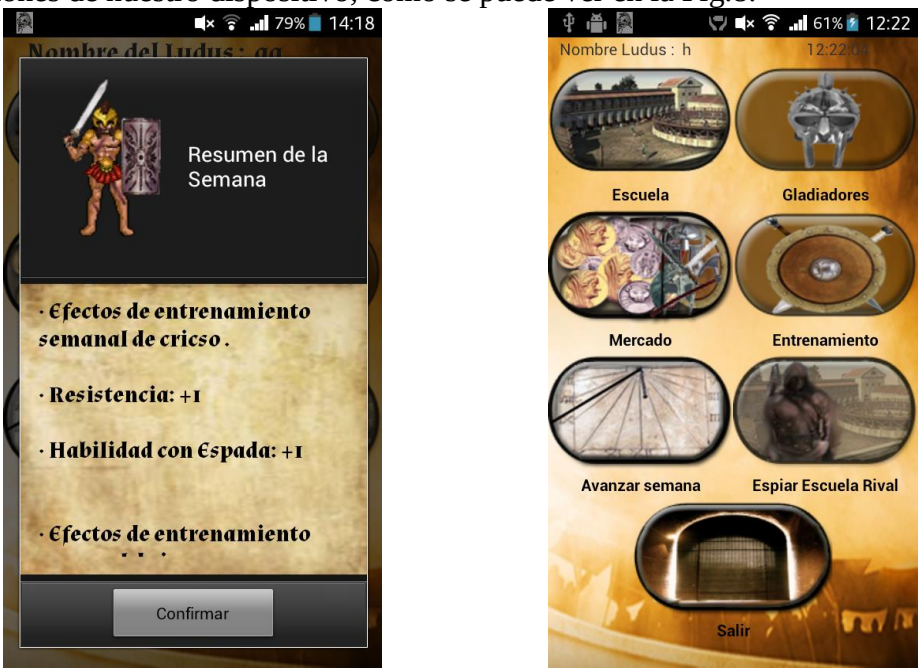


Fig. 8. Notificación de Combate

Si abrimos la barra de notificaciones y pulsamos sobre ella, nos aparecerá una descripción del combate, con el dinero del premio, inscripción, número de combates y lugar donde se celebrará. El usuario tendrá que decidir si participar o no en el combate. Si participa, podrá elegir si quiere mantener la equipación actual de su gladiador, o modificarla por una más conveniente. Una vez equipado, el gladiador del usuario y el del rival combatirán. La secuencia de interfaces cuando aparece un evento aparece en la *Fig.9.*

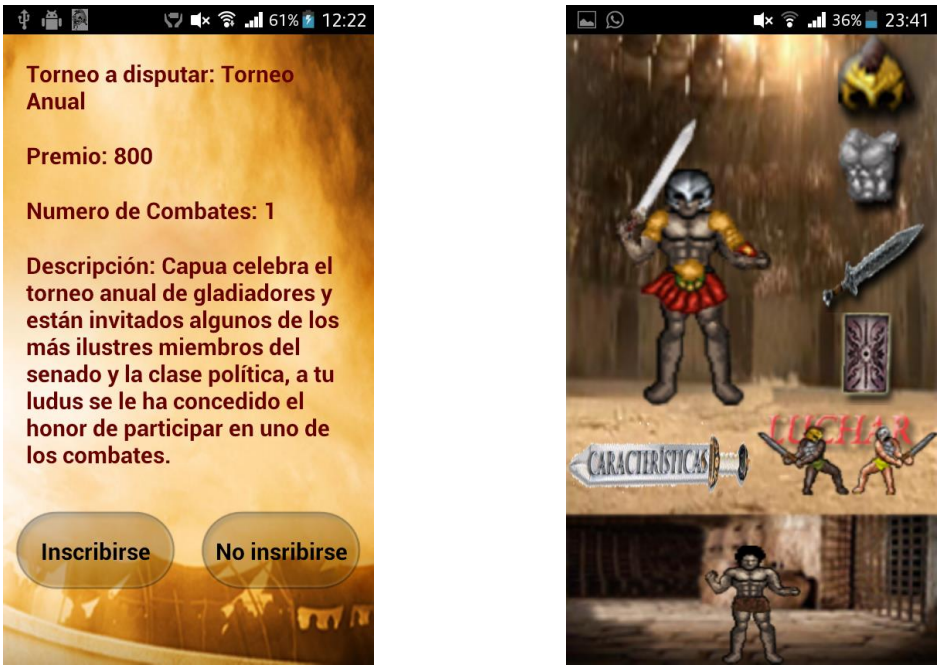


Fig.9. Notificación del combate y Pantalla de combate

Después de pulsar sobre el botón participar, podremos ver el desarrollo del combate mediante una secuencia de animaciones en dos dimensiones. El usuario no podrá interactuar con el gladiador, ya que son animaciones que representan, de modo gráfico, el resultado del algoritmo de combate. Para finalizar el combate, el usuario pulsará el botón "atrás" de su dispositivo. Dependiendo de si el resultado es una victoria, derrota o muerte, se mostrará una pantalla con un pequeño resumen de los premios recibidos, cantidad de vida restante o incremento de fama del gladiador.
En las Fig.10, Fig.11, y Fig.12 se muestran diversos ejemplos de la animación.

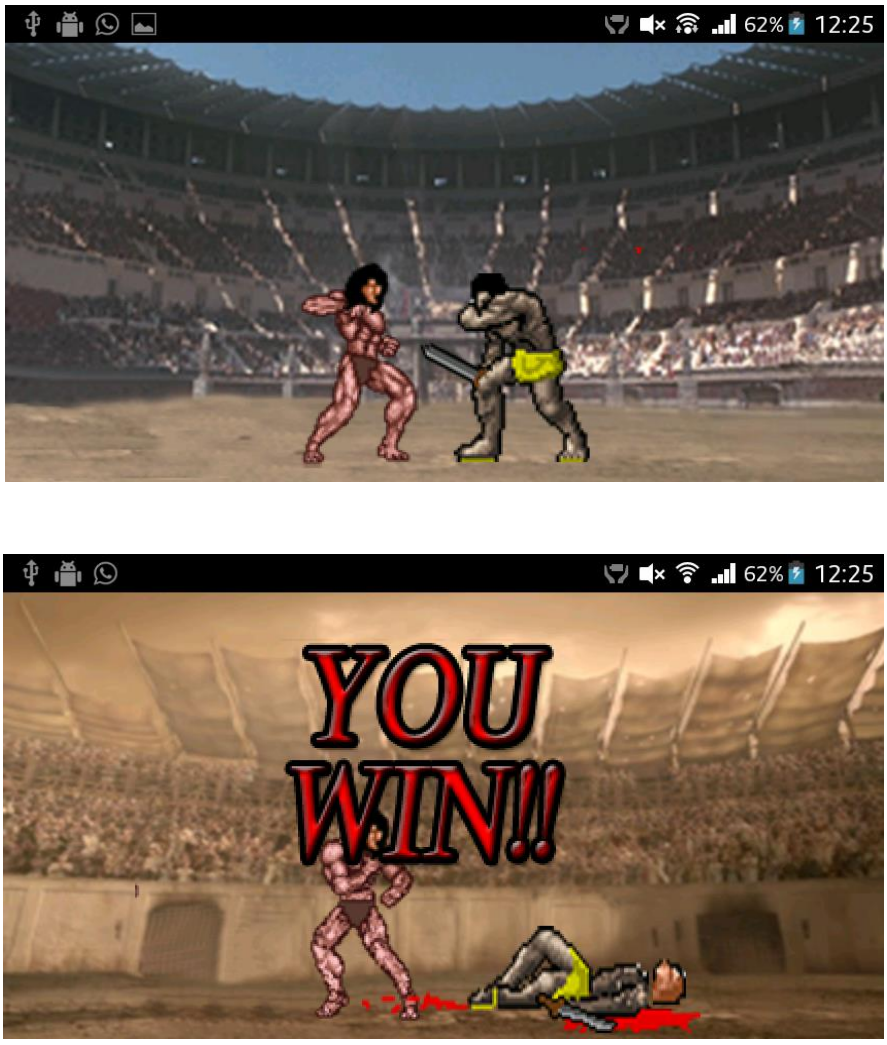


Fig.10. Muestra de la animación durante su ejecución

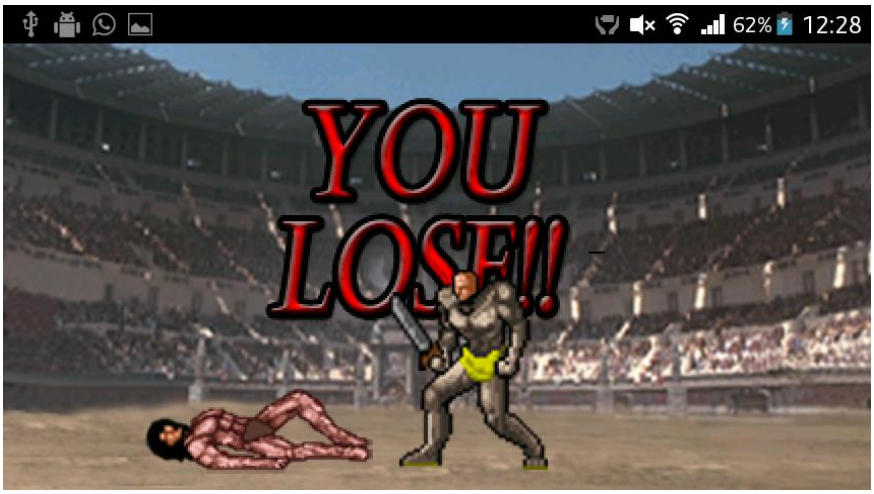
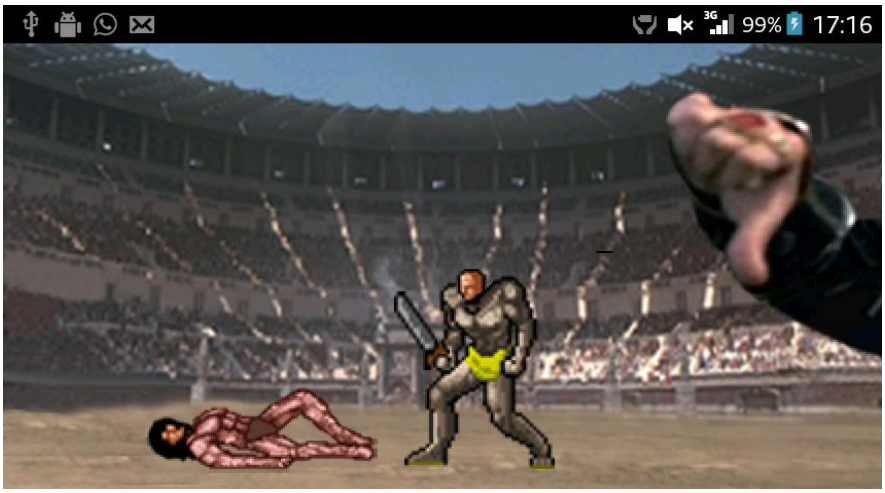


Fig. 11. Derrota durante el combate.

Si el evento tiene lugar en el Coliseo romano, habrá alguna ligera modificación en el desarrollo. En este caso nos enfrentaremos a un mayor número de rivales consecutivos y de mayor dificultad. En caso de derrota, será el propio César quien decida el destino de nuestro gladiador.



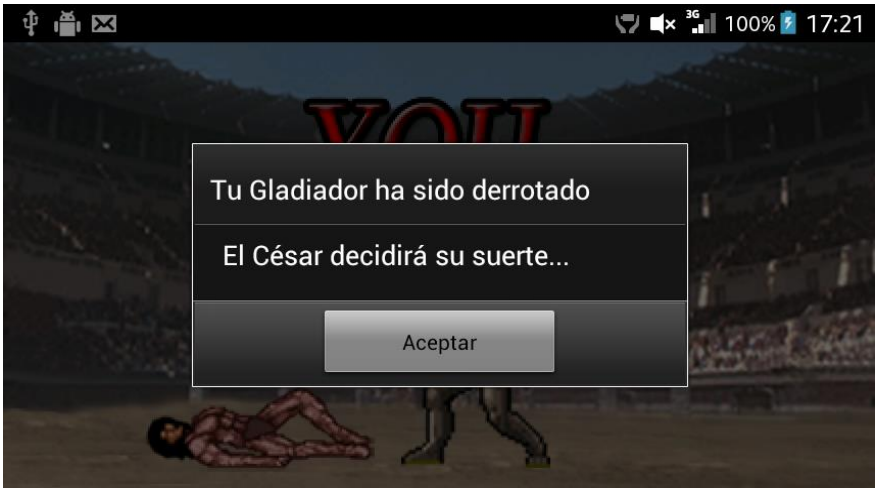


Fig. 12. Animación coliseo Romano

Por último, al usuario se le permitirá espiar al ludus rival para conocer el número de gladiadores y sus características, dinero, nivel o reputación, como se puede ver en la Fig.13. Eso sí, esto tendrá un coste económico que se deberá pagar cada vez que se quiera usar dicha opción. De esta forma el usuario podrá comprar o equipar a gladiadores de forma que puedan contrarrestar las habilidades del rival de una manera más efectiva.

Fig. 13. Espiar escuela rival

Si el usuario decide abandonar la partida, se mostrara un mensaje de confirmación como se muestra en la Fig.14. En caso afirmativo, se guardarán los cambios realizados, y en caso contrario, se seguirá en el menú principal.

Fig. 14. Botón salir